ANYTIME TRAINING WITH SCHEDULE-FREE SPECTRAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Standard neural network training relies on learning-rate schedules tied to a fixed horizon, leading to strong path dependence and costly re-tuning as data availability changes. Schedule-Free (SF) methods address this by removing explicit schedules, yet SF-AdamW, the current state-of-the-art anytime optimizer, consistently underperforms well-tuned AdamW baselines. We propose SF-NorMuon, a schedule-free spectral optimizer that closes this gap: with a single hyperparameter configuration, SF-NorMuon matches or exceeds tuned AdamW on 125M and 772M parameter language models across 1–8$\times$ Chinchilla horizons, and trails horizon-aware, cosine-scheduled NorMuon by only ~ 0.03 nats, substantially narrowing the gap between anytime and scheduled spectral optimizers. On the theoretical side, we prove a stationarity guarantee for schedule-free spectral dynamics and identify weight decay at the fast iterate as essential for long-horizon stability. SF-NorMuon enables practitioners to obtain high-quality checkpoints at any point during training without committing to a horizon in advance. Fully matching horizon-aware spectral optimizers without a schedule remains an open problem, which we highlight as a direction for future work. By closing the performance gap with tuned AdamW baselines, SF-NorMuon makes horizon-free optimization more practical, taking a step towards truly open-ended, continual learning.

1 INTRODUCTION

In the standard machine learning paradigm, after curating a dataset, a model is trained and then evaluated on a held-out, previously unseen portion of the data [30; 26; 7]. However, models are increasingly being deployed in scenarios where large amounts of data comparable to or even exceeding the training data are generated during the deployment period. Examples of this include large language models (LLMs) [9; 80; 89; 60] interacting with billions of users, and Vision-Language-Action (VLA) models controlling robots in industries [8; 58; 43]. As a result, judiciously leveraging the a priori unknown amount of data available at deployment time is crucial for improving model capabilities through continual learning [62; 83]. However, beyond addressing catastrophic forgetting [45; 19], there is also an optimization obstacle to overcome [18; 59; 17; 40].

Most existing training methods are not designed to be anytime, as they depend on learning rate schedules tied to a fixed training horizon and require extensive hyperparameter tuning under a predetermined compute budget [55; 31; 32]. For example, the standard cosine decay schedule has a strong path dependence. Consider training an LLaMA-2-style transformer with cosine decay as shown in Figure 1. The black dashed lines highlight the performance discrepancy: at the same token count, a run tuned for a shorter horizon significantly outperforms one tuned for a longer horizon, despite seeing identical data (e.g.: 2$\times$ vs 4$\times$ Chinchilla at 31B tokens for 772M model). This occurs because cosine decay ties the learning rate to the training horizon. At any intermediate point, runs with longer horizons have barely begun decaying, while shorter-horizon runs have already annealed to near zero, resulting in the big discrepancy in performances. Ideally, we want the model to learn as much as possible from a given set of training data regardless of how much data awaits in the future.

To that end, *Schedule-Free* (SF) methods [17] were recently proposed, removing the need for an explicit learning-rate schedule. The SF framework maintains three iterate sequences. In its most

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

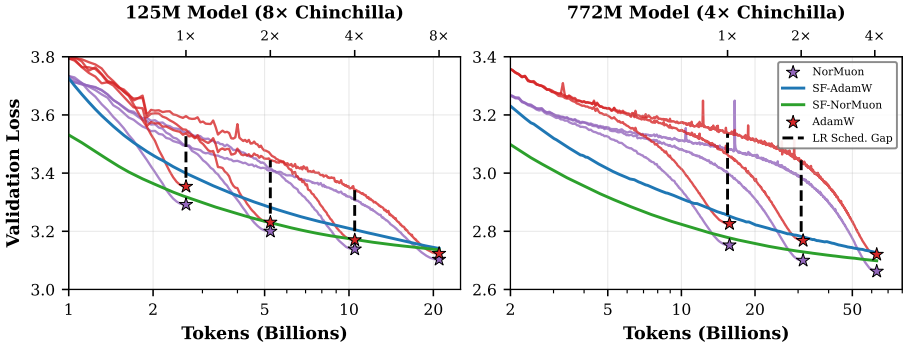

Figure 1: **Comparison of SF-NorMuon (this work), SF-AdamW, and *tuned* AdamW, and *tuned* NorMuon baselines for LLaMA-2-style transformers trained on FineWeb-100B.** *Left:* 125M model with AdamW learning rate tuned per horizon with cosine schedule. *Right:* 772M model with a single optimized AdamW configuration across horizons. Dashed lines highlight that learning rate schedule for a long horizon is sub-optimal for a smaller token budget. The grid of hyperparameters are summarized in Table 4, and validation losses for different learning rates are illustrated in Figure 6. SF-NorMuon consistently outperforms SF-AdamW, and matches *tuned* AdamW baselines, while horizon-aware NorMuon (cosine) remains ahead.

basic form, SF-SGD iterates as follows:

$$y_t = \beta x_t + (1-\beta) z_t, \tag{1}$$
$$z_{t+1} = z_t - \eta \nabla \mathcal{L}(y_t; \xi_t), \tag{2}$$
$$x_{t+1} = (1-c_{t+1}) x_t + c_{t+1} z_{t+1}, \tag{3}$$

where $z_1 = x_1$ and $c_{t+1} = 1/(t+1)$. The y sequence is where the gradient is evaluated at interpolated points, in spirit of Nesterov's accelerated method [61; 48], the z sequence is the *fast iterate* taking the (stochastic) gradient direction with a constant step size η throughout the training, and the x sequence is the *evaluation* sequence where validation loss is computed. Rewriting the update, we obtain

$$x_{t+1} = x_t - \eta c_{t+1} \nabla \mathcal{L}(y_t; \xi_t) + c_{t+1}(z_{t+1} - x_t). \tag{4}$$

Thus, even though z moves at a constant rate, the effective learning rate for x is $\eta_{\text{eff}} = \eta/(t+1)$, which decays over the duration of the training. Note that since x is a linear combination of y and x it can be computed on the fly, and so it does not need additional memory overhead. An alternative but closely related approach for horizon-free training is simply training with a constant learning rate, and returning an some moving average[1] of the weights [33; 51; 59; 34]. Notably, SF-AdamW, the practical variant of SF-SGD, won the first prize in the self-tuning track of the `AlgoPerf` challenge [41], demonstrating a new level of effectiveness for anytime training.

Despite this success, as observed in [29; 76; 71], SF-AdamW is unable to match the performance of well-tuned AdamW [44; 56], the primary workhorse for optimizing neural networks. This is also confirmed in Figure 1, where SF-AdamW (blue) consistently lies above the well-tuned AdamW baselines (red stars, tuning grid in Table 4) across all Chinchilla horizons. Since the schedule-free averaging mechanism in (1)–(3) is agnostic to the base update rule applied at (2), a natural question is whether a different choice of base optimizer can close this gap.

In modern neural networks, the vast majority of learnable parameters reside in weight matrices that act linearly on incoming activations. However, AdamW treats weight matrices as flat vectors, discarding how they act on activations. The spectral norm provides a natural measure of the effect a weight matrix update has on the output activation, motivating recent interest in spectral optimization methods such as Muon [36; 37] and its variants [74; 52; 2; 1; 4; 67; 3; 54; 42; 46]. Muon performs steepest descent under the spectral norm via the polar decomposition of the gradient computed by Newton-Schulz iterations, and has been successfully scaled to trillions of tokens [53; 78; 25; 75]. Neural network loss surfaces are characterized by many nearly-flat directions corresponding to small Hessian eigenvalues [68; 24], and thus taking uniformly sized steps in all spectral directions via the polar decomposition enables Muon to extract more information from the training data per step

[1] For instance, with $\beta = 0$ in (1), SF-SGD recovers constant SGD with uniform averaging.

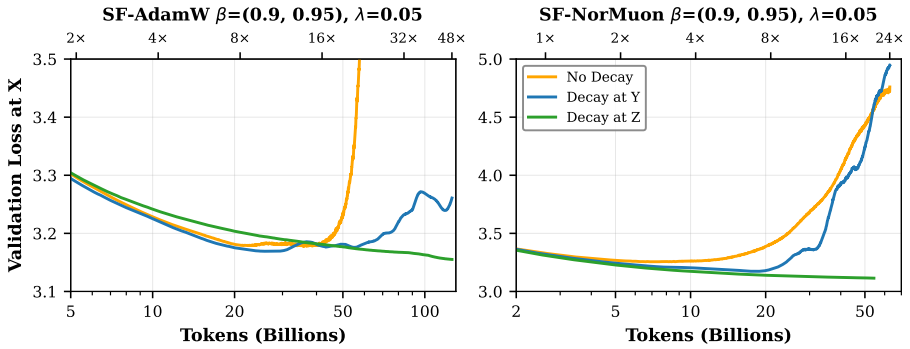

Figure 2: **Weight decay strategies for schedule-free optimizers.** *Left:* SF-AdamW with no decay (orange) diverges; decay at Y (blue) exhibits the best performance up to $\sim$30B tokens (c.f., Table 1), but eventually diverges. Decay at Z (green) yield stable training, but is suboptimal in the early phase. *Right:* SF-NorMuon with no decay (orange) diverges faster; decay at Y (blue) is also unstable. Decay at Z (green) maintains stable and performs the best throughout, motivating the analysis in Section 3.

[14]. Indeed, in our setting, cosine-scheduled NorMuon outperforms tuned AdamW at every horizon and both scales (Figure 1), making the spectral geometry a natural candidate base optimizer for schedule-free methods. This motivates the question:

Does optimization under the spectral norm improve upon existing schedule-free methods, and how close can it come to well-tuned, horizon-dependent AdamW and NorMuon baselines?

Apart from the choice of base geometry, schedule-free methods exhibit a stability challenge that becomes critical for long horizon training. As shown in Figure 2, the validation loss increases for long runs in absence of weight decay. This is in sharp contrast to the convex optimization theory underlying schedule-free methods, where convergence holds without regularization [17; 70]. Moreover, weight decay at y as originally proposed in [17], is inadequate for the spectral case, where the instability is significantly amplified (right panel of Figure 2). Therefore, in schedule-free methods weight decay plays a qualitatively different role: it is not merely a regularizer that improves generalization [47; 13; 65], but is *necessary* for long-horizon stability.

Combining these two insights, spectral geometry to speed up training and weight decay at the fast iterate to ensure stability, our contributions are as follows:

- We propose SF-SpectralSGD (with momentum), a schedule-free spectral method for matrix optimization, and prove a stationarity guarantee matching the $\widetilde{O}(T^{-1/4})$ rate achieved by recent work on convergence of Muon [10; 73; 69; 50] (Theorem 2.1 and Corollary 2.1).
- For practical deep learning, we incorporate the per-neuron normalization of NorMuon [52; 74] and obtain SF-NorMuon, as summarized in Algorithm 1. SF-NorMuon substantially improves over the state-of-the-art anytime optimizer SF-AdamW, matches tuned AdamW across training horizons, and comes within 0.025–0.036 nats of cosine-scheduled NorMuon, which has access to the horizon in advance (c.f., Figure 1, Table 1). Closing this remaining gap in a schedule-free manner remains an open problem. In particular, including explicit momentum buffer and per-neuron normalization is crucial in achieving good practical performance (c.f., Figure 3).
- We identify the key role of weight decay in schedule-free optimization. Unlike in standard optimizers, the fast iterate z_t in (2) continues to move at constant learning rate throughout training, so weight decay is a necessity for long-horizon stability. For spectral optimization, the aggressive polar updates amplify the instability (Figure 2), but weight decay directly applied to z_t leads to stable training. We prove boundedness of all iterates (Lemma 3.1), and derive a closed-form steady-state characterization (Lemma 3.2) that closely matches empirical training dynamics (Figure 4).

2 Schedule-Free Spectral Optimization

The fundamental design choice in a schedule-free optimizer is the geometry used for the fast update. When inputs and outputs are measured in the Euclidean norm $\|\cdot\|_2$, the induced operator norm on a

matrix $A \in \mathbb{R}^{m\times n}$ is $\|A\|_{2\to 2} := \sup_{\|u\|_2\leqslant 1} \|Au\|_2 = \sigma_{\max}(A) = \|A\|_{\mathrm{op}}$, where $\sigma_{\max}(A)$ denotes the largest singular value of A, which is precisely the spectral norm. Geometrically, it measures the maximum factor by which A can stretch any unit vector, making it the natural measure of how much a weight perturbation can alter the network's activations. The analysis for other induced norms, and their corresponding optimizers is presented in Appendix A.

The steepest-descent direction under a spectral-norm constraint solves [6; 64] $\operatorname{argmin}_{\|\Delta W\|_{\mathrm{op}}\leqslant\eta}\langle\nabla f(W), \Delta W\rangle_F$. Let $\nabla f(W) = U\Sigma V^\top$ be the SVD with $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$ orthonormal and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$. Decomposing $\Delta W = UQV^\top + R$ where R is orthogonal to the column/row space of the gradient, only Q contributes: $\langle\nabla f(W), \Delta W\rangle_F = \mathrm{tr}(\Sigma^\top Q) = \sum_{i=1}^r \sigma_i Q_{ii}$. Since $\|Q\|_{\mathrm{op}} \leqslant \eta$ implies $|Q_{ii}| \leqslant \eta$, the minimum is achieved at $Q^\star = -\eta I_r$. Thus, the polar factor is the steepest-descent direction under spectral norm: $\Delta W^\star = -\eta\, UV^\top = -\eta\,\mathrm{polar}(\nabla f(W))$.

Unlike Adam [44] or Lion [11] which act entry-wise and distort the gradient's singular-vector structure, the polar factor $UV^\top$ preserves the left and right singular subspaces. The polar update extracts learning signal from all singular directions of the gradient and not just those with the largest entries, and by making all the singular values unity, it makes uniform progress along all singular directions. This uniform treatment is more aggressive than coordinate-wise updates, so we include an explicit momentum buffer M_t to smooth the gradient before taking its polar factor. We build on this spectral-norm geometry to design a schedule-free optimizer that respects how weight matrices act on activations, while obtaining implicit learning-rate decay through online averaging.

2.1 Schedule-Free Spectral Descent with Momentum

For parameters $\beta, \mu \in [0, 1)$ and step-size $\eta > 0$, update rules for SF-Spectral Descent with Momentum are given by

$$\begin{aligned} &Y_t = \beta X_t + (1-\beta)Z_t, \qquad G_t = \nabla f(Y_t; \xi_t), \qquad M_t = \mu M_{t-1} + (1-\mu)G_t, \\ &P_t = \mathrm{polar}(M_t), \qquad Z_{t+1} = Z_t - \eta P_t, \qquad X_{t+1} = (1-c_{t+1})X_t + c_{t+1}Z_{t+1}, \end{aligned} \tag{5}$$

with $c_{t+1} = 1/(t+1)$, $X_1 = Z_1$ and $M_1 = G_1$. Here Z_t is the fast sequence with polar update, X_t is the returned schedule-free average, Y_t is the gradient-evaluation point, and M_t is an explicit momentum buffer.

Assume the existence of $W^\star \in \mathbb{R}^{m\times n}$ such that $f(W^\star) = f^\star$ and $\nabla f(W^\star) = 0$, and define $r := \min\{m, n\}$ and $\Delta := f(Y_1) - f^\star$.

Our convergence analysis relies on the following standard assumptions:

Assumption 2.1 (Frobenius Lipschitz smoothness)**.** *There is a constant $L_F > 0$ such that for all $U, V \in \mathbb{R}^{m\times n}$, $\|\nabla f(U) - \nabla f(V)\|_F \leqslant L_F \|U - V\|_F$.*

Assumption 2.2 (Unbiased gradient and bounded-variance noise)**.** *Let $\mathcal{F}_t$ be the sigma-field generated by all randomness up to the construction of Y_t. We assume, for some batch size $B \geqslant 1$ and $\sigma^2 \geqslant 0$,*

$$\mathbb{E}\big[G_t \mid \mathcal{F}_t\big] = \nabla f(Y_t), \qquad \mathbb{E}\Big[\|G_t - \nabla f(Y_t)\|_F^2 \mid \mathcal{F}_t\Big] \leqslant \sigma^2/B. \tag{6}$$

Assumption 2.3 (Bounded diameter)**.** *There exists a constant $D_F > 0$ such that $\|W^\star\|_F \leqslant D_F/2$ and $\|Z_t\|_F \leqslant D_F/2$ for all t.*

Note that bounded iterate assumption is typically employed in the convergence analysis of spectral methods (e.g., see [28, Theorem 7] and [4, Theorem 1]), and if necessary it can be enforced algorithmically by considering a projected variant that keep the fast iterates Z_t inside the corresponding norm ball. Further, with weight decay at Z, Assumption 2.3 can be lifted, as can be seen in Lemma 3.1.

Under these assumptions, we obtain the following convergence guarantee:

Theorem 2.1 (Stationarity of SF-Spectral Descent with Momentum)**.** *Under Assumption 2.1–2.3, for every $T \geqslant 1$, SF-Sepctral Descent with Momentum in (5) satisfies the following:*

$$\begin{aligned} \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant{}& \frac{\Delta + 2L_F D_F^2\big(\log(eT) + \pi^2/6\big)}{(1-\beta)T\eta} + \frac{2\sqrt{r}\,\sigma}{(1-\beta)\sqrt{B}}\left(\sqrt{\frac{1-\mu}{1+\mu}} + \frac{1}{(1-\mu)T}\right) \\ &+ \frac{2\sqrt{r}\,L_F D_F \log(eT)}{(1-\beta)T}\left(\frac{1+\mu}{1-\mu}\right) + \frac{L_F r\,\eta}{(1-\beta)}\left(\frac{1+3\mu}{1-\mu}\right). \end{aligned} \tag{7}$$

Algorithm 1 Schedule-Free NorMuon (SF-NorMuon)

Input: base learning rate η, interpolation parameter β_1, variance parameter β_2, momentum parameter μ, perturbation parameter ε, weight decay λ, warmup steps T_{warmup}
for $t = 1$ **to** T **do**
  $Y_t \leftarrow (1-\beta_1)Z_t + \beta_1 X_t$ ▷ Interpolate weights
  $G_t \leftarrow \nabla_{\mathbf{W}} \mathcal{L}(Y_t, \zeta_t)$ ▷ Compute gradient at interpolation point
  $M_t \leftarrow \mu M_{t-1} + (1-\mu) G_t$ ▷ Explicit momentum buffer

  $P_t \leftarrow \text{polar}(M_t)$ ▷ Polar factor of momentum
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)\,\text{mean}_{\text{cols}}(P_t \odot P_t)$ ▷ Row-wise second moment
  $V_t \leftarrow \text{ExpandRows}(v_t)$ ▷ Broadcast to matrix
  $\widehat{P}_t \leftarrow P_t \oslash (\sqrt{V_t} + \varepsilon)$ ▷ Adaptive row-wise normalization

  $\eta_t \leftarrow \eta \min(1, t/T_{\text{warmup}})$ ▷ Learning rate with warmup
  $\hat{\eta}_t \leftarrow 0.2\eta_t \sqrt{mn}/\|\widehat{P}_t\|_F$ ▷ Match Adam RMS scaling
  $s_t \leftarrow s_{t-1} + \eta_t^2$
  $c_{t+1} \leftarrow \eta_t^2 / s_t$ ▷ Averaging coefficient

  $Z_{t+1} \leftarrow Z_t - \eta\lambda Z_t - \hat{\eta}_t \widehat{P}_t$ ▷ Update with weight decay at Z_t
  $X_{t+1} \leftarrow (1-c_{t+1})X_t + c_{t+1} Z_{t+1}$ ▷ Update averaged iterate
end for
Return: X_T

The first term is the descent term, the next captures the stochastic noise, the third term is the price of schedule-free drift, and the final term reflects the constant spectral step and momentum tracking. The explicit momentum stabilizes the aggressive polar steps at the cost of a tracking error. By optimizing the terms that appear on the right hand side of the theorem above, we obtain the following corollary for any $\beta \in [0, 1)$, both with and without noise.

Corollary 2.1 (Optimized convergence rates for SF-Spectral Descent)**.** *Under Assumption 2.1–2.3, for any fixed $\beta \in [0, 1)$ there exist choices $\mu_T \in [0, 1)$ and $\eta_T > 0$ such that*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla f(Y_t)\|_* = \widetilde{O}(T^{-1/4}) \ \ (\sigma > 0); \ \ \textit{and} \ \ \frac{1}{T}\sum_{t=1}^{T} \|\nabla f(Y_t)\|_* = \widetilde{O}(T^{-1/2}) \ \ (\sigma = 0). \quad (8)$$

These convergence rates match the ones derived in recent works on convergence of Muon [10; 73; 69; 50]. An analogous result can also be proven under Lipschitz smoothness measured in the spectral norm rather than the Frobenius norm. Since the Frobenius-smooth version is more directly comparable to existing matrix-optimization methods, we state it here and defer the spectral-smooth variant, together with the full proofs of both results, to the Appendix C.

2.2 Schedule-Free NorMuon for Neural Network Training

To make schedule-free spectral descent practical for neural network training, we incorporate three refinements below. The resulting algorithm is SF-NorMuon, which we summarize in Algorithm 1. A PyTorch implementation of this algorithm is provided in Appendix G.

Learning rate warm up and scaling. We warm up the learning rate linearly, which stabilizes early training [27] and enables larger learning rates [38]. After computing the adaptively normalized direction $\widehat{P}_t$, the algorithm scales the base learning rate by $\hat{\eta}_t = 0.2\eta_t\sqrt{\max(m,n)}/\|\widehat{P}_t\|_F$. This scaling serves two purposes [53]: (i) It normalizes the effective step size to be comparable to Adam's RMS-normalized updates, enabling η and λ to be shared with SF-AdamW; and (ii) it provides additional stability by accounting for the overall magnitude of the normalized update direction.

Weight decay at the fast iterate Z. Weight decay is applied to Z_t, rather than Y_t (as originally proposed in [16] for SF-AdamW). This choice treats weight decay as a modification to the base optimizer dynamics rather than as an ℓ_2 regularizer in the loss, which would correspond to decay at Y_t. The scaled weight decay term $\eta_t\lambda Z_t$ is subtracted from Z_t in the base update. Weight decay at Z_t is necessary for long-horizon training with schedule-free methods, as we explain in Section 3.

Row-wise Normalization. We incorporate row-wise adaptive normalization [52; 74]. The polar factor can produce orthogonalized updates with high variance in step sizes across individual neurons, as different rows of a weight matrix may have vastly different update magnitudes. To accomodate, SF-NorMuon maintains a running average $v_t \in \mathbb{R}^m$ of the squared row norms of the polar factor P_t, capturing the typical magnitude of updates for each neuron and enabling per-neuron step size adaptation. The exponential moving average with parameter β_2 provides stability while remaining responsive to changes in gradient statistics, analogous to the second-moment estimation in Adam, but operating at the neuron level rather than element-wise.

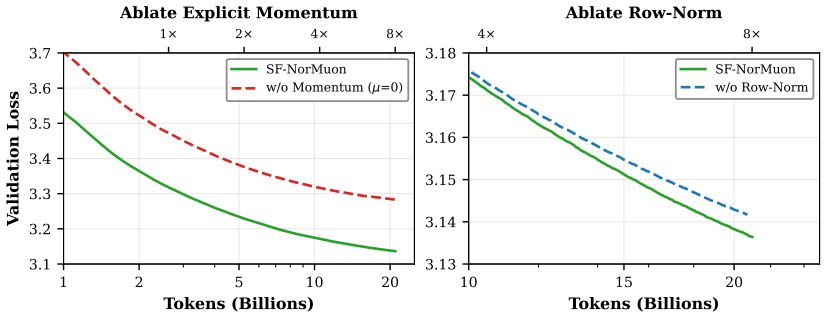

Figure 3: **Importance of explicit momentum and row-wise normalization for SF-NorMuon.** *Left:* Ablating explicit momentum ($\mu = 0$) significantly degrades performance, with final loss increasing from 3.14 to 3.28. This validates the importance of smoothing the gradient before computing the polar factor, as discussed in Section 2. *Right:* Ablating row-wise normalization leads to a smaller gap, with SF-NorMuon reaching the same loss approximately 12% faster.

In Figure 3, we ablate the two key aspects of SF-NorMuon: (i) the explicit momentum buffer (by setting $\mu = 0$), which reduces to direct polar decomposition of the gradient, and (ii) the row-wise adaptive normalization (by removing the v_t computation and normalization step), which reduces to uniform spectral steps. We find that removing the row-wise adaptive normalization slows down convergence by around 12% at 8x Chinchilla ratio, but removing momentum leads to a significant drop in performance. Note that the original SF-SGD or SF-AdamW does not have an explicit momentum buffer. Therefore, the above ablation study illustrates how direct application of schedule-free dynamics to spectral optimzer (e.g., replacing $\nabla\mathcal{L}(y_t;\xi_t)$ in (2) with polar$(\nabla\mathcal{L}(y_t;\xi_t))$) would perform.

Remark 1 (Memory footprint)**.** *Consider a matrix parameter of shape $m \times n$. For SF methods, we need to track Y and Z (since X can be computed on the fly). For SF-AdamW the second moment buffer (per parameter) takes up memory mn, whereas SF-NorMuon uses a row-wise second-moment buffer of size m. Thus, there is room for an explicit momentum buffer matrix M of size $m \times n$ buffer unlike SF-AdamW. Hence, the total persistent memory is $3mn$ for SF-AdamW and $3mn + m$ for SF-NorMuon, compared with $3mn$ for AdamW and $2mn + m$ for NorMuon.*

3 Necessity of Weight Decay in Long Horizon Training

Modern language models are trained for very long horizons [72; 49; 23], and the training tokens for Mixture-of-Experts models can exceed 30 times the Chinchilla ratio to the number of active parameters [25; 75; 78]. Thus, to assess long-horizon behavior of schedule-free methods, we train the 125M LLaMa-2-style model for up to 48× Chinchilla with the default hyper-parameters.

Interestingly, for SF methods, the validation loss at X diverges without weight decaying *specifically at Z*. This is in sharp contrast to convex optimization theory, where convergence holds without regularization [17; 70]. In particular, for SF-AdamW in Figure 2 (left panel), decay at Y performs the best up to around 30B tokens, but eventually leads to divergence. In contrast, for SF-NorMuon in Figure 2 (right panel), decaying at Z is not only stable (see also Lemma 3.1) but also maintains the best validation loss throughout the long training exceeding 50B tokens.

To better understand this phenomenon, we consider a quasi steady-state analysis, following [82; 15]. Recall from (4) that the schedule-free method can be rewritten as

$$X_{t+1} = X_t - c_{t+1}\eta U_t + c_{t+1}(Z_{t+1} - X_t), \tag{9}$$

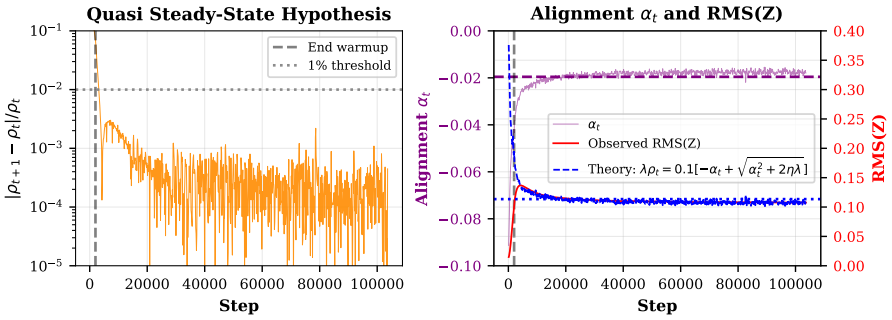

Figure 4: **Quasi steady-state analysis for SF-NorMuon with decay at Z (averaged over layers).** *Left:* The ratio $|\rho_{t+1} - \rho_t|/\rho_t$ remains below 1% after warmup, validating the quasi steady-state hypothesis (Hypothesis 3.1). *Right:* Alignment α_t (purple) and RMS(Z) (red) over training. The theoretical prediction $\lambda\rho_t = 0.1[-\alpha_t + \sqrt{\alpha_t^2 + 2\eta\lambda}]$ from Lemma 3.2 with $\kappa = 0.2$ (blue dashed) closely tracks the observed values. This training run corresponds to $\eta = 0.01, \lambda = 0.05$.

where $c_{t+1} = 1/(t+1)$, and U_t is either gradient divided by second moment buffer for SF-AdamW or polar transform of momentum buffer for SF-NorMuon, respectively. As can be seen, the evaluation sequence X_t evolves with an *effective* learning rate $\eta_t^{\text{eff}} = c_{t+1}\eta = \eta/(t+1)$ that decays to zero as $t \to \infty$, which makes the divergence surprising.

The culprit is in the fast Z_t sequence: it evolves with constant learning rate η throughout training, and in fact, we observe in practice that validation loss at Z_t often diverges. Since X_t averages all past $\{Z_s\}_{s \leqslant t}$, Z_t eventually contaminates the X_t sequence.

For SF-NorMuon, applying weight decay directly at Z_t addresses this issue, while maintaining good performance. To make this precise, we prove that weight decay at Z_t ensures all iterates to remain bounded, and characterize the precise steady-state norm to which Z_t converges.

Lemma 3.1. *Consider the update $Z_{t+1} \leftarrow Z_t - \eta\lambda Z_t - \hat{\eta}_t \widehat{P}_t$ and assume $0 < \eta\lambda < 1$ with $\hat{\eta} = \kappa\,\eta\sqrt{mn}/\|\widehat{P}_t\|_F$ for a constant $\kappa > 0$. Then, we have for all t,*

$$\|Z_t\|_F \leqslant \|Z_0\|_F(1-\eta\lambda)^t + \frac{\kappa\sqrt{mn}}{\lambda} \implies \|Z_t\|_F \leqslant \frac{\kappa\sqrt{mn}}{\lambda} \text{ as } t \to \infty. \tag{10}$$

Furthermore, since X_t and Y_t are convex combinations of $\{Z_s\}_{s \leqslant t}$, they satisfy the same bound.

We make the following remark regarding the divergence of weight decay at Y_t in Figure 2:

Remark 2. *For weight decay at Y_t, the coupled dynamics (Z_t, D_t) where $D_t = Z_t - X_t$ satisfy:*

$$\begin{bmatrix} \|Z_{t+1}\|_F \\ \|D_{t+1}\|_F \end{bmatrix} \leqslant \begin{bmatrix} 1-\eta\lambda & \eta\lambda\beta \\ (1-c_{t+1})\eta\lambda & (1-c_{t+1})(1+\eta\lambda\beta) \end{bmatrix} \begin{bmatrix} \|Z_t\|_F \\ \|D_t\|_F \end{bmatrix} + \kappa\,\eta\sqrt{mn} \begin{bmatrix} 1 \\ 1-c_{t+1} \end{bmatrix}. \tag{11}$$

In the long-horizon limit $c_t \to 0$, the iteration matrix above has the leading eigenvalue

$$\lambda_1 = 1 + \frac{\eta\lambda}{2}\left(\sqrt{1+6\beta+\beta^2} - (1-\beta)\right) > 1 \tag{12}$$

for all $\beta > 0$, and so the triangle-inequality upper bound similar to (10) diverges.

Having established iterate boundedness in Lemma 3.1, we now study how the norm of Z evolves during the training. For this purpose, the analysis greatly simplifies if we make a quasi steady-state hypothesis following [82; 15], which states that change in norm of Z between two iterations is much smaller than the norm itself.

Hypothesis 3.1 (Quasi Steady-State)**.** *There exists an iteration count τ and a constant $\varepsilon \geqslant 0$ such that for all $t \geqslant \tau$, $|\|Z_{t+1}\|_F - \|Z_t\|_F| \leqslant \varepsilon\|Z_t\|_F$.*

To enable uniform comparison across layers of different shapes, we define the RMS norm $\rho_t := \|Z_t\|_F/\sqrt{mn}$ for an $m \times n$ weight matrix, and the alignment $\alpha_t := \langle \widehat{P}_t, Z_t\rangle/(\|\widehat{P}_t\|_F\|Z_t\|_F) \in [-1, 1]$, which is the cosine of the angle between the preconditioned update and the current iterate. The left panel of Figure 4 shows that $|\rho_{t+1} - \rho_t|/\rho_t$ remains small after warmup, validating the hypothesis. Based on this, we derive the steady-state value of ρ_t :

Lemma 3.2 (Quasi Steady-State of Z). *Consider the update $Z_{t+1} \leftarrow Z_t - \eta\lambda Z_t - \hat{\eta}_t \widehat{P}_t$ and assume $0 < \eta\lambda \ll 1$ with $\hat{\eta} = \kappa\,\eta\sqrt{mn}/\|\widehat{P}_t\|_F$ for a constant $\kappa > 0$. Then, the steady-state satisfies*

$$\lambda\rho_t = \frac{\kappa}{2}\left[-\alpha_t + \sqrt{\alpha_t^2 + 2\eta\lambda}\right]. \tag{13}$$

The constant κ enters both lemmas only as a multiplicative factor, so the results hold for any RMS-matching choice, including the exact factor matching AdamW's update RMS; our experiments use the heuristic value $\kappa = 0.2$ from [53].

After the end of the warm-up period, the alignment α_t approaches a small and negative value that persists for the rest of the training period as shown by the purple curve in right panel of Figure 4. Based on the Lemma 3.2, we expect that the RMS norm of Z (depicted in red) will asymptote to a constant value, which is also confirmed by the red curve sitting on top of the theoretical prediction shown in blue. Crucially, since the Z norm reaches a fixed value, the fast iterate remains controlled throughout training, and stabilizes other sequences as shown in Lemma 3.1.

4 Experiments on Training Language Models

Table 1: **Validation loss at X across Chinchilla horizons.** *Speedup* (in **bold**) is the percentage of steps saved by SF-NorMuon to reach the final loss of SF-AdamW. SF methods use hyperparameters tuned at $2\times$ Chinchilla for both models. AdamW uses cosine schedule tuned per-horizon (c.f., Table 4) for 125M model, and the configuration for $8\times$ was reused for the 772M model. SF-AdamW uses weight decay at Y as it performs the best in these horizons (c.f., Figure 2) and is the originally proposed option in [17]. NorMuon (cosine) is a horizon-aware scheduled reference, tuned per-horizon for the 125M model and with a single configuration reused across horizons for the 772M model.

	125M Model					772M Model				
Chinchilla Ratio	SF-NorMuon (decay@Z)	SF-AdamW (decay@Y)	AdamW (cosine)	NorMuon (cosine)	Speedup	SF-NorMuon (decay@Z)	SF-AdamW (decay@Y)	AdamW (cosine)	NorMuon (cosine)	Speedup
$1\times$	3.318	3.399	3.354	3.292	**35%**	2.777	2.852	2.826	2.752	**50%**
$2\times$	3.229	3.282	3.231	3.200	**35%**	2.729	2.780	2.768	2.700	**52%**
$4\times$	3.170	3.205	3.170	3.139	**36%**	2.698	2.726	2.720	2.662	**47%**
$8\times$	3.136	3.141	3.124	3.103	**11%**	—	—	—	—	—

We validate SF-NorMuon on auto-regressive language modeling by training decoder only transformers at two scales: a 125M parameter model and a 772M parameter model. Both architectures follow the LLaMA-2 design [81], with tied input-output embeddings, rotary positional embeddings (RoPE) [77], RMSNorm for pre-normalization [86], and squared ReLU activations in the MLP blocks [88]. Additional architecture details are provided in Appendix D. Our training code is based on the NanoGPT repository [39; 35].

Dataset and training setup. We train on the FineWeb-100B dataset [63] tokenized with the GPT-2 tokenizer [66]. All experiments use batch size of 512 sequences and sequence length of 1024 tokens, yielding approximately 524K tokens per gradient step. We adopt the Chinchilla scaling [31], where $N\times$ Chinchilla denotes training on $\sim 20N$ tokens per model parameter. For our 125M model, this corresponds to 2.5B tokens at $1\times$ (5,000 steps), scaling up to 21B tokens at $8\times$ (40,000 steps).

Hyperparameter tuning. We start with the 125M model. For schedule-free optimizers, we use 2,000 warm up steps and tune hyper-parameters at $2\times$ Chinchilla, and reuse these for all settings. The best hyperparameters for SF-AdamW are $\eta = 0.01$, $(\beta_1, \beta_2) = (0.95, 0.99)$, and $\lambda = 0.05$. For SF-NorMuon, we reuse the same learning rate and weight decay parameter, following the $0.2\eta_t\sqrt{mn}/\|\widehat{P}_t\|_F$ rescaling factor from [78]. The other hyper-parameters were set to $(\beta_1, \beta_2) = (0.9, 0.95)$, momentum $\mu = 0.8$. For the non-matrix parameters (embeddings, layer norms), both schedule-free methods run SF-AdamW. For the AdamW baseline, we tune the learning rate *separately at each horizon*, using 2,500 warm up steps followed by cosine decay to zero; comparison of these runs can be found in Figure 6 in Appendix D. Other hyperparameters were tuned at $2\times$ Chinchilla. This represents a well-tuned baseline with per-horizon optimization, using optimized $\eta \in \{0.004, 0.008\}$, and $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\lambda = 0.1$.

For the 772M parameter model, we reuse the optimal $8\times$ Chinchilla hyper-parameters from AdamW and the $2\times$-tuned hyper-parameters for schedule-free methods. This approach is motivated by the

observation that losses near the optimal learning rates exhibit only small variation, and recent work has shown similar optimal learning rates across small and large Llama-2 model scales (see Table 7 and Table 35 in [71]). For scheduled NorMuon on the 772M model, we use a single configuration across horizons ($\eta = 0.004$, $\lambda = 0.1$, $\mu = 0.95$, $\beta_2 = 0.95$, 2,000 warm up steps, cosine decay to zero). We omit the $8\times$ Chinchilla runs for the large model due to compute limitations.

Results. Table 1 reports validation losses (evaluated at X) across training horizons. For both models, SF-NorMuon with weight decay at Z consistently outperforms SF-AdamW across all horizons. Crucially, SF-NorMuon matches the per-horizon tuned AdamW baselines (except at $8\times$ Chinchilla where it lags marginally), while SF-AdamW lags behind consistently. This demonstrates that SF-NorMuon delivers competitive performance with AdamW at any stopping point, without a horizon-specific learning rate schedule. For the 772M model, the speedup advantage of SF-NorMuon over SF-AdamW is even more pronounced: 50% at $1\times$ and 52% at $2\times$ Chinchilla.

Comparison with scheduled NorMuon. We also report NorMuon with a cosine schedule (Table 1, Figure 1). Scheduled NorMuon outperforms all other methods at both scales, and SF-NorMuon trails it by 0.025–0.036 nats. This gap is expected, since NorMuon knows the horizon in advance and benefits from a final learning-rate cooldown [85], whereas SF-NorMuon relies only on implicit decay through online averaging. Matching horizon-aware NorMuon with a truly schedule-free method remains an open problem. We additionally compare against NorMuon with a warmup-stable-decay (WSD) schedule [32; 33] in Appendix E.2: WSD trails cosine at every horizon and, at $2\times$ Chinchilla, is outperformed by SF-NorMuon as well. A constant-learning-rate NorMuon with EMA weights [87], which uses the same $\sim 3mn$ memory as SF-NorMuon (Remark 1), also remains behind SF-NorMuon (Appendix E.3). All reported gaps are several times larger than the seed-to-seed standard deviation ($\leqslant 0.006$ nats; Appendix E.4), and SF-NorMuon incurs only $\sim 2\%$ wall-clock overhead per step relative to AdamW (Appendix G).

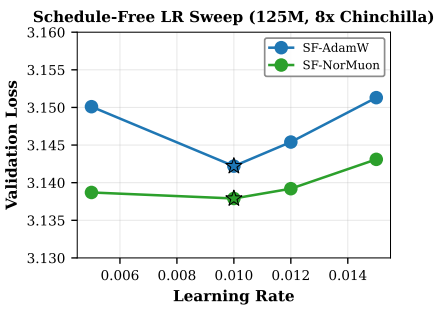

Figure 5: Learning rate sweep comparison between SF-AdamW and SF-NorMuon.

Improvement across learning rates. A key practical advantage of SF-NorMuon is robustness to the choice of learning rate. Figure 5 shows a learning-rate sweep at $8\times$ Chinchilla for both schedule-free methods: SF-NorMuon achieves lower validation loss than SF-AdamW across all learning rates tested. Importantly, SF-NorMuon's performance degrades gracefully away from the optimum, confirming that the spectral geometry provides a favorable optimization landscape even without careful per-run tuning.

Other findings. Full convergence proofs are in Appendix C. Additional material includes derivations of spectral and sign updates (Appendix A), a review of SF-SGD and SF-AdamW (Appendix B), architectures and hyperparameter search spaces (Appendix D), ablations, WSD/EMA comparisons, and seed variability (Appendix E), ViT-B/16 experiments on ImageNet-1K, where SF-NorMuon again outperforms SF-AdamW and tuned AdamW at every milestone (Appendix F), and a PyTorch implementation (Appendix G).

5 Conclusion

In this work, we proposed SF-NorMuon, a schedule-free spectral optimizer that matches per-horizon tuned AdamW across Chinchilla horizons without requiring schedule or horizon specification in advance. We validated SF-NorMuon on both auto-regressive language modeling (125M and 772M parameter LLaMA-2-style models) and vision transformers (ViT-B/16 on ImageNet-1K), demonstrating consistent improvements over SF-AdamW. We proved a $\widetilde{O}(T^{-1/4})$ stationarity guarantee for the underlying schedule-free spectral dynamics for smooth non-convex functions and identified weight decay at the fast iterate as essential for long-horizon stability, which is a departure from the convex optimization theory. Extending the validation of SF-NorMuon to larger scales and multi-stage continual learning pipelines is a natural direction for future work. Since weight decay at the fast iterate drives the weight norm to a steady state (Lemma 3.2), combining schedule-free spectral updates with explicitly norm-constrained training [84] is another promising direction.

REFERENCES

[1] Kwangjun Ahn, Noah Amsel, and John Langford. Dion2: A simple method to shrink matrix in muon, 2025. URL https://arxiv.org/abs/2512.16928.

[2] Kwangjun Ahn, Byron Xu, Natalie Abreu, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint: 2504.05295*, 2025.

[3] Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm, 2026. URL https://arxiv.org/abs/2505.16932.

[4] Kang An, Yuxing Liu, Rui Pan, Yi Ren, Shiqian Ma, Donald Goldfarb, and Tong Zhang. Asgo: Adaptive structured gradient optimization, 2025. URL https://arxiv.org/abs/2503.20762.

[5] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '24, pp. 929–947. ACM, April 2024. doi: 10.1145/3620665.3640366. URL http://dx.doi.org/10.1145/3620665.3640366.

[6] Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology, 2024. URL https://arxiv.org/abs/2409.20325.

[7] Christopher M. Bishop and Hugh Bishop. *Deep Learning: Foundations and Concepts*. Springer International Publishing, 2024. ISBN 9783031454684. doi: 10.1007/978-3-031-45468-4. URL http://dx.doi.org/10.1007/978-3-031-45468-4.

[8] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] Da Chang, Yongxiang Liu, and Ganzhao Yuan. On the convergence of muon and beyond, 2026. URL https://arxiv.org/abs/2509.15816.

[11] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023.

[12] Jack Choquette, Wishwesh Gandhi, Olivier Giroux, Nick Stam, and Ronny Krashinsky. Nvidia a100 tensor core gpu: Performance and innovation. *IEEE Micro*, 41(2):29–35, March 2021. ISSN 1937-4143. doi: 10.1109/mm.2021.3061394. URL http://dx.doi.org/10.1109/MM.2021.3061394.

[13] Francesco D'Angelo, Maksym Andriushchenko, Aditya Varre, and Nicolas Flammarion. Why do we need weight decay in modern deep learning?, 2024. URL https://arxiv.org/abs/2310.04415.

[14] Damek Davis and Dmitriy Drusvyatskiy. When do spectral gradient updates help in deep learning?, 2026. URL https://arxiv.org/abs/2512.04299.

[15] Aaron Defazio. Why gradients rapidly increase near the end of training, 2025. URL https://arxiv.org/abs/2506.02285.

[16] Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by d-adaptation. In *International Conference on Machine Learning*, pp. 7449–7479. PMLR, 2023.

[17] Aaron Defazio, Xingyu Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. *Advances in Neural Information Processing Systems*, 37:9974–10007, 2024.

[18] Thomas Degris, Khurram Javed, Arsalan Sharifnassab, Yuxin Liu, and Richard Sutton. Step-size optimization for continual learning. *arXiv preprint arXiv:2401.17401*, 2024.

[19] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. ISSN 1939-3539. doi: 10.1109/tpami.2021.3057446. URL http://dx.doi.org/10.1109/TPAMI.2021.3057446.

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

[22] Aleksandr Dremov, Alexander Hägele, Atli Kosson, and Martin Jaggi. Training dynamics of the cooldown stage in warmup-stable-decay learning rate scheduler, 2025. URL https://arxiv.org/abs/2508.01483.

[23] William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23 (120):1–39, 2022.

[24] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2232–2241. PMLR, 2019. URL https://proceedings.mlr.press/v97/ghorbani19b.html.

[25] GLM-5-Team. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[27] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[28] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization, 2018. URL https://arxiv.org/abs/1802.09568.

[29] Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna B Allal, Leandro Von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. *Advances in Neural Information Processing Systems*, 37:76232–76264, 2024.

[30] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2009. ISBN 9780387848587. doi: 10.1007/978-0-387-84858-7. URL http://dx.doi.org/10.1007/978-0-387-84858-7.

[31] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

[32] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies, 2024. URL https://arxiv.org/abs/2404.06395.

[33] Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations, 2024. URL https://arxiv.org/abs/2405.18392.

[34] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[35] Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, @fernbear.bsky.social, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and @Grad62304977. modded-nanogpt: Speedrunning the nanogpt baseline, 2024. URL https://github.com/KellerJordan/modded-nanogpt.

[36] Keller Jordan et al. Muon: An optimizer for hidden layers in neural networks. https://kellerjordan.github.io/posts/muon/, 2024. Accessed: 2026-01-25.

[37] Keller Jordan et al. Muon (github repository): An optimizer for hidden layers in neural networks, 2024. URL https://github.com/KellerJordan/Muon. GitHub repository, master branch.

[38] Dayal Singh Kalra and Maissam Barkeshli. Why warmup the learning rate? underlying mechanisms and improvements. *Advances in Neural Information Processing Systems*, 37: 111760–111801, 2024.

[39] Andrej Karpathy. NanoGPT. https://github.com/karpathy/nanoGPT, 2022.

[40] Priya Kasimbeg, Vincent Roulet, Naman Agarwal, Sourabh Medapati, Fabian Pedregosa, Atish Agarwala, and George E Dahl. How far away are truly hyperparameter-free learning algorithms? *arXiv preprint arXiv:2505.24005*, 2025.

[41] Priya Kasimbeg, Frank Schneider, Runa Eschenhagen, Juhan Bae, Chandramouli Shama Sastry, Mark Saroufim, Feng Boyuan, Less Wright, Edward Z. Yang, Zachary Nado, Sourabh Medapati, Philipp Hennig, Michael Rabbat, and George E. Dahl. Accelerating neural network training: An analysis of the AlgoPerf competition. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CtM5xjRSfm.

[42] Ahmed Khaled, Kaan Ozkara, Tao Yu, Mingyi Hong, and Youngsuk Park. Muonbp: Faster muon via block-periodic orthogonalization, 2025. URL https://arxiv.org/abs/2510.16981.

[43] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024.

[44] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

[45] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

[46] Alexey Kravatskiy, Ivan Kozyrev, Nikolai Kozlov, Alexander Vinogradov, Daniil Merkulov, and Ivan Oseledets. The ky fan norms and beyond: Dual norms and combinations for matrix optimization, 2025. URL https://arxiv.org/abs/2512.09678.

[47] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Proceedings of the 5th International Conference on Neural Information Processing Systems*, NIPS'91, pp. 950–957, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. ISBN 1558602224.

[48] Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1):365–397, 2012.

[49] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

[50] Jiaxiang Li and Mingyi Hong. A note on the convergence of muon, 2025. URL https://arxiv.org/abs/2502.02900.

[51] Yunshui Li, Yiyuan Ma, Shen Yan, Chaoyi Zhang, Jing Liu, Jianqiao Lu, Ziwen Xu, Mengzhao Chen, Minrui Wang, Shiyi Zhan, Jin Ma, Xunhao Lai, Deyi Liu, Yao Luo, Xingyan Bin, Hongbin Ren, Mingji Han, Wenhao Hao, Bairen Yi, LingJun Liu, Bole Ma, Xiaoying Jia, Xun Zhou, Siyuan Qiao, Liang Xiang, and Yonghui Wu. Model merging in pre-training of large language models, 2025. URL https://arxiv.org/abs/2505.12082.

[52] Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. Normuon: Making muon more efficient and scalable, 2025. URL https://arxiv.org/abs/2510.05491.

[53] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training, 2025. URL https://arxiv.org/abs/2502.16982.

[54] Ziyue Liu, Ruijie Zhang, Zhengyang Wang, Yequan Zhao, Yupeng Su, Zi Yang, and Zheng Zhang. Muon^2: Boosting muon via adaptive second-moment preconditioning, 2026. URL https://arxiv.org/abs/2604.09967.

[55] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

[56] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[57] Jianhao Ma and Yuxin Chen. Wsqd: A horizon-free learning rate schedule for large model training, 2026. URL https://arxiv.org/abs/2607.10959.

[58] Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai, 2024.

[59] Alexandru Meterez, Pranav Ajit Nair, Depen Morwani, Cengiz Pehlevan, and Sham Kakade. Anytime pretraining: Horizon-free learning-rate schedules with weight averaging. *arXiv preprint arXiv:2602.03702*, 2026.

[60] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey, 2024.

[61] Yurii Nesterov. A method for solving the convex programming problem with convergence rate o (1/k2). In *Dokl akad nauk Sssr*, volume 269, pp. 543, 1983.

[62] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, May 2019. ISSN 0893-6080. doi: 10.1016/j.neunet.2019.01.012. URL http://dx.doi.org/10.1016/j.neunet.2019.01.012.

[63] Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

[64] Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos, 2025. URL https://arxiv.org/abs/2502.07529.

[65] Shikai Qiu, Zixi Chen, Hoang Phan, Qi Lei, and Andrew Gordon Wilson. Hyperparameter transfer enables consistent gains of matrix-preconditioned optimizers across scales, 2026. URL https://arxiv.org/abs/2512.05620.

[66] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. 2019.

[67] Liliang Ren, Yang Liu, Yelong Shen, and Weizhu Chen. Rethinking language model scaling under transferable hypersphere optimization, 2026. URL https://arxiv.org/abs/2603.28743.

[68] Levent Sagun, Utku Evci, V. Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks, 2017.

[69] Naoki Sato, Hiroki Naganuma, and Hideaki Iiduka. Convergence bound and critical batch size of muon optimizer, 2025. URL https://arxiv.org/abs/2507.01598.

[70] Fabian Schaipp, Alexander Hägele, Adrien Taylor, Umut Simsekli, and Francis Bach. The surprising agreement between convex optimization theory and learning-rate scheduling for large model training. *arXiv preprint arXiv:2501.18965*, 2025.

[71] Andrei Semenov, Matteo Pagliardini, and Martin Jaggi. Benchmarking optimizers for large language model pretraining, 2026. URL https://openreview.net/forum?id=Jw7khYzYzl.

[72] Noam Shazeer, Azalia Mirhoseini, Krzysztof Mahdavi, Andrew Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

[73] Wei Shen, Ruichuan Huang, Minhui Huang, Cong Shen, and Jiawei Zhang. On the convergence analysis of muon, 2026. URL https://arxiv.org/abs/2505.23737.

[74] Chongjie Si, Debing Zhang, and Wei Shen. Adamuon: Adaptive muon optimizer, 2025. URL https://arxiv.org/abs/2507.11005.

[75] Varun Singh, Lucas Krauss, Sami Jaghouar, Matej Sirovatka, Charles Goddard, Fares Obied, Jack Min Ong, Jannik Straube, Fern, Aria Harley, Conner Stewart, Colin Kealty, Maziyar Panahi, Simon Kirsten, Anushka Deshpande, Anneketh Vij, Arthur Bresnu, Pranav Veldurthi, Raghav Ravishankar, Hardik Bishnoi, DatologyAI Team, Arcee AI Team, Prime Intellect Team, Mark McQuade, Johannes Hagemann, and Lucas Atkins. Arcee trinity large technical report, 2026. URL https://arxiv.org/abs/2602.17004.

[76] Minhak Song, Beomhan Baek, Kwangjun Ahn, and Chulhee Yun. Through the river: Understanding the benefit of schedule-free methods for language model training. In *High-dimensional Learning Dynamics 2025*, 2025. URL https://openreview.net/forum?id=b5HYeRzG9M.

[77] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

[78] Kimi Team. Kimi k2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

[79] Kimi Team. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

[80] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

[81] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

[82] Twan van Laarhoven. L2 regularization versus batch and weight normalization, 2017. URL https://arxiv.org/abs/1706.05350.

[83] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024. URL https://arxiv.org/abs/2302.00487.

[84] Kaiyue Wen, Xingyu Dang, Kaifeng Lyu, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them ii: Hyperball optimization, 2026. URL https://arxiv.org/abs/2606.16899.

[85] Kaichao You, Mingsheng Long, Jianmin Wang, and Michael I. Jordan. How does learning rate decay help modern neural networks?, 2019. URL https://arxiv.org/abs/1908.01878.

[86] Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019. URL https://arxiv.org/abs/1910.07467.

[87] Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean Foster, and Sham Kakade. How does critical batch size scale in pre-training?, 2025. URL https://arxiv.org/abs/2410.21676.

[88] Zhengyan Zhang, Yixin Song, Guanghui Yu, Xu Han, Yankai Lin, Chaojun Xiao, Chenyang Song, Zhiyuan Liu, Zeyu Mi, and Maosong Sun. Relu^2 wins: Discovering efficient activation functions for sparse llms, 2024. URL https://arxiv.org/abs/2402.03804.

[89] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2023.

TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL FOR "ANYTIME TRAINING WITH SCHEDULE-FREE SPECTRAL OPTIMIZATION"

In this appendix, we provide more thorough preliminaries and background information that supplement the main text. We also state the formal version of the informal theorems from the main text, as well as their proofs. This appendix is organized as follows:

- In Appendix A, we review metrized deep learning, and derive optimizers from norms on weight matrices.
- In Appendix B , we review the Schedule-free method. In particular, we discuss SF-SGD and its convergence bound, and the algorithm for SF-AdamW.
- In Appendix C proves in the detail the convergence of SF-Spectral Descent, for noisy Lipschitz smooth functions in both the Frobenius and spectral norm. In addition, we prove the lemmas used in the steady-state analysis.
- In Appendix D, we review the architecture of the language models used in our experiments and present the hyper-parameter selection procedure.
- In Appendix E, studies the effect of ablating momentum term for SF-NorMuon (setting $\mu = 0$), and ablating the row-normalization. We also compare against NorMuon with a WSD schedule and with EMA weights, and report seed-to-seed variability.
- In Appendix F, we train ViT-B/16 on ImageNet-1K and report the architecture, training setup, and hyperparameter sweeps.
- In Appendix G, we present a PyTorch implementation of SF-NorMuon algorithm, together with a wall-clock comparison of all optimizers.

A OPTIMIZERS FROM WEIGHT-SPACE MATRIX GEOMETRY

For the reader's convenience, we collect here the basic derivations of the update rules discussed in the main text. The starting point is always the same: given a current iterate W_t and gradient $G_t := \nabla f(W_t)$, we choose an update ΔW_t by minimizing the first-order model of the loss subject to a constraint, or equivalently a quadratic penalty, in the geometry of interest. Our treatment follows the work of Bernstein and Newhouse [6].

A.1 STEEPEST DESCENT IN A PRESCRIBED GEOMETRY

Let $\|\cdot\|_{\mathcal{X}}$ be any norm on the space of matrices. The corresponding norm-constrained steepest-descent step is

$$\Delta W_t \in \arg\min_{\|\Delta W\|_{\mathcal{X}} \leqslant \eta} \langle G_t, \Delta W \rangle. \tag{14}$$

Since the objective is linear, the solution lies on the boundary of the ball and points in an extremal descent direction for the chosen norm. Writing

$$\|G_t\|_{\mathcal{X},*} := \sup_{\|U\|_{\mathcal{X}} \leqslant 1} \langle G_t, U \rangle \tag{15}$$

for the dual norm, (14) is equivalent to

$$\Delta W_t = -\eta\, U_t, \qquad U_t \in \arg\max_{\|U\|_{\mathcal{X}} \leqslant 1} \langle G_t, U \rangle, \tag{16}$$

and the optimal value is $-\eta \|G_t\|_{\mathcal{X},*}$.

In practice one often uses the equivalent regularized local model

$$\Delta W_t \in \arg\min_{\Delta W} \left\{ \langle G_t, \Delta W \rangle + \frac{1}{2\eta} \|\Delta W\|^2_{\mathcal{X},t} \right\}, \tag{17}$$

where $\|\cdot\|_{\mathcal{X},t}$ may depend on time through a momentum estimate or a preconditioner. The constrained form (14) produces normalized directions such as sign and polar updates, while the regularized form (17) produces the preconditioned-gradient forms used by practical optimizers.

A.2 THE SIGN GEOMETRY: $p = 1$, $q = \infty$

Take the matrix norm

$$\|A\|_{\ell_1 \to \ell_\infty} = \sup_{\|u\|_1 \leqslant 1} \|Au\|_\infty. \tag{18}$$

A direct calculation shows that this is simply the entrywise max norm:

$$\|A\|_{\ell_1 \to \ell_\infty} = \max_{i,j} |A_{ij}|. \tag{19}$$

Indeed, for each row i,

$$|(Au)_i| = \left| \sum_j A_{ij} u_j \right| \leqslant \max_j |A_{ij}| \sum_j |u_j| \leqslant \max_j |A_{ij}|, \tag{20}$$

so $\|Au\|_\infty \leqslant \max_{i,j} |A_{ij}|$. Equality is achieved by choosing u to be a signed coordinate vector supported on an entry attaining the maximum.

Substituting (19) into (14) gives

$$\min_{\max_{i,j} |\Delta W_{ij}| \leqslant \eta} \sum_{i,j} (G_t)_{ij} (\Delta W)_{ij}. \tag{21}$$

This decouples coordinatewise, and for each entry the minimizer is

$$(\Delta W_t)_{ij} = -\eta \operatorname{sign}\big((G_t)_{ij}\big). \tag{22}$$

Hence the steepest-descent rule is

$$\Delta W_t = -\eta \operatorname{sign}(G_t), \tag{23}$$

which is the base update underlying signSGD:

$$W_{t+1} = W_t - \eta \operatorname{sign}(G_t). \tag{24}$$

A.3 MOMENTUM IN THE SIGN GEOMETRY

To overcome the noise from the computing the gradient on a mini-batch a standard practical modification is to replace the raw gradient by a smoothed momentum estimate. Let

$$M_t = \beta M_{t-1} + (1 - \beta) G_t, \qquad 0 \leqslant \beta < 1. \tag{25}$$

Applying the same steepest-descent rule to M_t instead of G_t yields

$$W_{t+1} = W_t - \eta \operatorname{sign}(M_t), \tag{26}$$

which is the update for signSGD with momentum.

Lion. Lion uses a sign update applied to a momentum direction that directly incorporates current gradient as well. In the standard implementation, one first forms

$$C_t = \beta_1 M_{t-1} + (1 - \beta_1) G_t, \tag{27}$$

then updates

$$W_{t+1} = W_t - \eta \operatorname{sign}(C_t), \tag{28}$$

and finally refreshes the state

$$M_t = \beta_2 M_{t-1} + (1 - \beta_2) G_t. \tag{29}$$

Thus Lion can be viewed as a two-timescale version of the basic sign steepest-descent rule.

Adam with $\beta_1 = \beta_2 = 0$. Adam augments the gradient with both a momentum buffer and a variance buffer:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) G_t, \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2) G_t^{\odot 2}, \tag{30}$$

where $G_t^{\odot 2}$ denotes the entrywise square. Its coordinatewise update is

$$W_{t+1} = W_t - \eta \frac{m_t}{\sqrt{v_t} + \varepsilon}. \tag{31}$$

When $\beta_1 = \beta_2 = 0$, both buffers become memoryless,

$$m_t = G_t, \qquad v_t = G_t^{\odot 2}, \tag{32}$$

and so, ignoring ε,

$$W_{t+1} = W_t - \eta \operatorname{sign}(G_t). \tag{33}$$

Thus, memoryless Adam recovers signSGD exactly [6]. Note that with $\beta_2 = 0$ but $\beta_1 > 0$ the update is $m_t/(|G_t| + \varepsilon)$, which is *not* $\operatorname{sign}(m_t)$ in general; signSGD with momentum is instead obtained by applying the sign steepest-descent rule directly to the momentum buffer, as derived above.

A.4 The spectral geometry: $p = q = 2$

Now take the matrix norm

$$\|A\|_{\ell_2 \to \ell_2} = \|A\|_{\mathrm{op}}, \tag{34}$$

the spectral norm. The norm-constrained step becomes

$$\Delta W_t \in \arg \min_{\|\Delta W\|_{\mathrm{op}} \leqslant \eta} \langle G_t, \Delta W \rangle. \tag{35}$$

Let the singular value decomposition of G_t be

$$G_t = U_t \Sigma_t V_t^\top. \tag{36}$$

By von Neumann's trace inequality,

$$\langle G_t, U \rangle \leqslant \|G_t\|_* \|U\|_{\mathrm{op}} \qquad \text{for all } U, \tag{37}$$

so over the unit spectral-norm ball the maximum of $\langle G_t, U \rangle$ is $\|G_t\|_*$, attained by

$$U = U_t V_t^\top. \tag{38}$$

This matrix is the polar factor of G_t, which we denote by $\operatorname{polar}(G_t)$. Therefore

$$\Delta W_t = -\eta \operatorname{polar}(G_t), \tag{39}$$

and the corresponding update for spectral descent is

$$W_{t+1} = W_t - \eta \operatorname{polar}(G_t). \tag{40}$$

A.5 Momentum in the spectral geometry

Muon. Exactly as in the sign case, we may replace G_t by a momentum-smoothed matrix

$$M_t = \beta M_{t-1} + (1-\beta) G_t. \tag{41}$$

Applying spectral steepest descent to M_t gives the update for Muon

$$W_{t+1} = W_t - \eta \operatorname{polar}(M_t). \tag{42}$$

In practical implementations, this polar factor is not computed by an exact SVD but is approximated by a small number of Newton-Schulz iterations. This minimizes the computational overhead as they can be implemented using only matrix multiplications (GEMMs).

Shampoo. Shampoo maintains left and right second-moment buffers for each matrix parameter and updates by preconditioning with their inverse quarter-powers

$$M_t = \beta_1 M_{t-1} + (1-\beta_1) G_t, \qquad L_t = \beta_2 L_{t-1} + (1-\beta_2) M_t M_t^\top, \qquad R_t = \beta_2 R_{t-1} + (1-\beta_2) M_t^\top M_t, \tag{43}$$

followed by the update

$$W_{t+1} = W_t - \eta\, L_t^{-1/4} M_t R_t^{-1/4}. \tag{44}$$

In the limiting case $\beta_2 = 0$, these buffers collapse to the instantaneous matrices

$$L_t = M_t M_t^\top, \qquad R_t = M_t^\top M_t. \tag{45}$$

If $M_t = U\Sigma V^\top$ is the singular value decomposition, then

$$L_t^{-1/4} M_t R_t^{-1/4} = U\Sigma^{-1/2} U^\top (U\Sigma V^\top) V \Sigma^{-1/2} V^\top = UV^\top = \mathrm{polar}(M_t), \tag{46}$$

where the identity is exact in the idealized full-rank case (and otherwise understood with pseudo-inverses on the support of M_t). Hence, when the second-moment buffers are collapsed in this way, Shampoo reduces to the same polar transform of the momentum matrix that underlies spectral descent and Muon.

B Review of SF-SGD and SF-AdamW

Schedule-free (SF) methods were introduced to remove the need for a prescribed training horizon in learning-rate scheduling. The basic idea is to keep a *fast* sequence z_t, on which the base optimizer runs at a nearly constant learning rate, while returning an *averaged* sequence x_t whose effective learning rate decays automatically through online averaging. Gradients are evaluated at an interpolation point y_t between these two states. In this way, schedule-free optimization replaces explicit time-based annealing by implicit annealing through online weight averaging [17].

SF-SGD. In its simplest form, Schedule-Free SGD maintains three sequences,

$$y_t = \beta x_t + (1-\beta) z_t, \qquad g_t = \nabla \mathcal{L}(y_t; \xi_t), \qquad z_{t+1} = z_t - \eta g_t, \tag{47}$$

together with the online average

$$x_{t+1} = (1 - c_{t+1}) x_t + c_{t+1} z_{t+1}, \qquad c_{t+1} = \frac{1}{t+1}. \tag{48}$$

Here z_t is the base iterate, x_t is the returned iterate, and y_t is the point at which the stochastic gradient is computed. The parameter $\beta \in [0,1]$ interpolates between Polyak-Ruppert averaging ($\beta = 0$) and primal averaging ($\beta = 1$).

Implicit decay of the effective learning rate. The averaging update can be rewritten as

$$x_{t+1} - x_t = c_{t+1}(z_{t+1} - x_t) = c_{t+1}(z_t - x_t) - c_{t+1}\eta g_t. \tag{49}$$

Using

$$y_t = \beta x_t + (1-\beta) z_t \qquad \Longrightarrow \qquad z_t - x_t = \frac{y_t - x_t}{1-\beta}, \tag{50}$$

we obtain

$$x_{t+1} - x_t = \frac{c_{t+1}}{1-\beta}(y_t - x_t) - c_{t+1}\eta g_t. \tag{51}$$

Thus the returned iterate x_t moves as though it were taking a gradient step with *effective* learning rate

$$\eta_t^{\mathrm{eff}} \approx c_{t+1}\eta = \frac{\eta}{t+1}. \tag{52}$$

Thus, the effective learning rate decays like $1/t$ even though the base update on z_t uses a constant step size. This is the basic mechanism by which schedule-free methods replace an explicit schedule. In the convex Lipschitz setting, this implicit learning rate decay is sufficient to retain the optimal worst-case convergence rate of $O(T^{-1/2})$. For a convex G-Lipschitz function with $D = \|x_1 - x_\star\|$, the choice of $\eta = D/(G\sqrt{T})$ for any $\beta \in [0,1]$ leads to

$$\mathbb{E}\big[F(x_T) - F(x_\star)\big] \leqslant \frac{DG}{\sqrt{T}}. \tag{53}$$

Algorithm 2 Schedule-Free AdamW

Input: base learning rate η, interpolation parameter β_1, variance parameter β_2, weight decay λ, warmup steps T_{warmup}, numerical constant ε, decay location $\tilde{y}_t \in \{y_t, z_t\}$
for $t = 1, 2, \ldots, T$ **do**
  $y_t \leftarrow (1-\beta_1)z_t + \beta_1 x_t$ ▷ Momentum via interpolation
  $g_t \leftarrow \nabla \mathcal{L}(y_t; \xi_t)$ ▷ Gradient is evaluated at y_t
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^{\odot 2}$
  $\eta_t \leftarrow \eta \sqrt{1-\beta_2^t}\, \min(1, t/T_{\text{warmup}})$ ▷ Warmup and Adam bias-correction
  $z_{t+1} \leftarrow z_t - \eta_t\, g_t/(\sqrt{v_t}+\varepsilon) - \eta_t \lambda \tilde{y}_t$
  $s_t \leftarrow s_{t-1} + \eta_t^2$
  $c_{t+1} \leftarrow \eta_t^2 / s_t$
  $x_{t+1} \leftarrow (1-c_{t+1})x_t + c_{t+1} z_{t+1}$ ▷ Update weighted iterate average
end for
return x_T

SF-AdamW. For deep-learning applications, the raw gradient is replaced by a preconditioned update together with warmup and decoupled weight decay.

The choice $\tilde{y}_t = y_t$ corresponds to applying weight decay at the gradient location, which matches the interpretation of decay as an ℓ_2-regularizer in the loss, while $\tilde{y}_t = z_t$ applies decay at the fast iterate. A useful practical feature is that SF-AdamW does not require more memory than AdamW. One stores z_t and y_t in memory, and reconstructs

$$x_t = \frac{y_t - (1-\beta_1)z_t}{\beta_1} \tag{54}$$

when evaluation is needed. Note that unlike classical momentum, which introduces a separate momentum buffer, schedule-free momentum is implemented through the interpolation $y_t = (1-\beta)z_t + \beta x_t$ in weight space.

C Convergence of Schedule-Free Spectral optimization

We first state the iterations of the schedule-free spectral optimization and the assumptions of Lipschitz smoothness and bounded diameter.

Iterates. For parameters $0 \leqslant \beta < 1$, $0 \leqslant \mu < 1$, stepsize $\eta > 0$, and batch size $B \geqslant 1$, consider

$$Y_t = \beta X_t + (1-\beta) Z_t, \tag{55}$$

$$G_t = \frac{1}{B} \sum_{i=1}^{B} \nabla f(Y_t; \xi_{t,i}), \tag{56}$$

$$M_0 = G_0, \qquad M_t = \mu M_{t-1} + (1-\mu) G_t \quad (t \geqslant 1), \tag{57}$$

$$P_t = \operatorname{polar}(M_t), \tag{58}$$

$$Z_{t+1} = Z_t - \eta P_t, \tag{59}$$

$$X_{t+1} = \frac{t}{t+1} X_t + \frac{1}{t+1} Z_{t+1}, \qquad X_0 = Z_0. \tag{60}$$

Assume there exists $W^\star \in \mathbb{R}^{m \times n}$ such that

$$f(W^\star) = f^\star, \qquad \nabla f(W^\star) = 0,$$

and let

$$r := \min\{m, n\}, \qquad \Delta := f(Y_0) - f^\star.$$

Throughout, the stochastic oracle is unbiased with bounded Frobenius variance:

$$\mathbb{E}[G_t \mid \mathcal{F}_t] = \nabla f(Y_t) =: g_t, \qquad \mathbb{E}\big[\|G_t - g_t\|_F^2 \mid \mathcal{F}_t\big] \leqslant \frac{\sigma^2}{B}. \tag{61}$$

Smoothness assumptions. For the spectral proof, we assume there exists a constant L_S such that

$$\|\nabla f(A) - \nabla f(B)\|_* \leqslant L_S \|A - B\|_{\mathrm{op}} \qquad \text{for all } A, B \in \mathbb{R}^{m\times n}. \tag{62}$$

For the Frobenius proof, we assume there exists a constant L_F such that

$$\|\nabla f(A) - \nabla f(B)\|_F \leqslant L_F \|A - B\|_F \qquad \text{for all } A, B \in \mathbb{R}^{m\times n}. \tag{63}$$

Bounded diameter assumptions. For the spectral proof, we assume there exists a constant $D_S > 0$ such that

$$\|W^\star\|_{\mathrm{op}} \leqslant D_S/2, \qquad \|Z_t\|_{\mathrm{op}} \leqslant D_S/2 \qquad \text{for all } t, \tag{64}$$

while for the Frobenius proof, we assume there exists a constant $D_F > 0$ such that

$$\|W^\star\|_F \leqslant D_F/2, \qquad \|Z_t\|_F \leqslant D_F/2 \qquad \text{for all } t. \tag{65}$$

Consequently, $\|X_t\|_{\mathrm{op}}, \|Y_t\|_{\mathrm{op}} \leqslant D_S/2$ for all t, and hence

$$\|Y_t - W^\star\|_{\mathrm{op}} \leqslant D_S, \qquad \|Z_t - W^\star\|_{\mathrm{op}} \leqslant D_S \qquad \text{for all } t,$$

and a similar statement holds for the Frobenius norm.

These are standard bounded-iterate assumptions (see Theorem 7 in [28] and Theorem 1 in [4]), and they can be enforced algorithmically by considering a projected variant that keep the fast iterates Z_t inside the corresponding norm ball. The iterates X, Y are unweighted and weighted averages of current and previous Z iterates and hence remain within the ball.

C.1 Reusable identities and bounds

Lemma C.1 (Iterate identities)**.** *Define*

$$S_t := Z_t - Y_t.$$

Then $S_t = \beta(Z_t - X_t)$, and for every $t \geqslant 0$,

$$S_{t+1} = \frac{t}{t+1}\big(S_t - \beta\eta P_t\big), \tag{66}$$

$$Y_{t+1} - Y_t = \frac{S_t}{t+1} - \eta\left(1 - \beta + \frac{\beta}{t+1}\right) P_t. \tag{67}$$

Consequently, if $\|\cdot\|_\diamond$ is any norm for which $\|Y_t - W^\star\|_\diamond \leqslant D_\diamond$ and $\|Z_t - W^\star\|_\diamond \leqslant D_\diamond$, then

$$\|S_t\|_\diamond \leqslant 2D_\diamond, \qquad \|Y_{t+1} - Y_t\|_\diamond \leqslant \frac{2D_\diamond}{t+1} + \eta\|P_t\|_\diamond. \tag{68}$$

In particular,

$$\|Y_{t+1} - Y_t\|_{\mathrm{op}} \leqslant \frac{2D_S}{t+1} + \eta, \tag{69}$$

$$\|Y_{t+1} - Y_t\|_F \leqslant \frac{2D_F}{t+1} + \eta\sqrt{r}. \tag{70}$$

Proof. Since $Y_t = \beta X_t + (1-\beta)Z_t$, we have

$$S_t = Z_t - Y_t = \beta(Z_t - X_t).$$

Using (59) and (60),

$$\begin{aligned}
S_{t+1} &= Z_{t+1} - Y_{t+1} \\
&= \beta(Z_{t+1} - X_{t+1}) \\
&= \beta\left(Z_{t+1} - \frac{t}{t+1}X_t - \frac{1}{t+1}Z_{t+1}\right) \\
&= \frac{t}{t+1}\beta(Z_{t+1} - X_t) \\
&= \frac{t}{t+1}\beta\big((Z_t - \eta P_t) - X_t\big) \\
&= \frac{t}{t+1}\big(S_t - \beta\eta P_t\big),
\end{aligned}$$

which proves (66). Then

$$\begin{aligned} Y_{t+1} - Y_t &= (Z_{t+1} - S_{t+1}) - (Z_t - S_t) \\ &= -\eta P_t - S_{t+1} + S_t \\ &= \frac{S_t}{t+1} - \eta \left(1 - \beta + \frac{\beta}{t+1}\right) P_t, \end{aligned}$$

which is (67). The norm bounds follow from

$$\|S_t\|_\diamond = \|Z_t - Y_t\|_\diamond \leqslant \|Z_t - W^\star\|_\diamond + \|Y_t - W^\star\|_\diamond \leqslant 2D_\diamond,$$

together with $\|P_t\|_{\mathrm{op}} \leqslant 1$ and $\|P_t\|_F \leqslant \sqrt{r}$. □

Lemma C.2 (Polar alignment)**.** *For every $t \geqslant 0$,*

$$\langle g_t, P_t \rangle \geqslant \|g_t\|_* - 2\|g_t - M_t\|_*. \tag{71}$$

Proof. Since $P_t = \mathrm{polar}(M_t)$,

$$\langle M_t, P_t \rangle = \|M_t\|_*.$$

Also,

$$\langle g_t - M_t, P_t \rangle \geqslant -\|g_t - M_t\|_* \|P_t\|_{\mathrm{op}} \geqslant -\|g_t - M_t\|_*.$$

Therefore

$$\langle g_t, P_t \rangle = \langle M_t, P_t \rangle + \langle g_t - M_t, P_t \rangle \geqslant \|M_t\|_* - \|g_t - M_t\|_*.$$

Finally,

$$\|M_t\|_* \geqslant \|g_t\|_* - \|g_t - M_t\|_*,$$

which yields (71). □

Lemma C.3 (Stochastic EMA error)**.** *Let*

$$C_0 = g_0, \qquad C_t = \mu C_{t-1} + (1-\mu) g_t, \qquad E_t := C_t - M_t.$$

Then

$$E_t = \mu E_{t-1} + (1-\mu)(g_t - G_t), \tag{72}$$

and

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\|E_t\|_* \leqslant \sqrt{r} \left(\frac{\sigma}{(1-\mu)T\sqrt{B}} + \sqrt{\frac{1-\mu}{1+\mu}} \frac{\sigma}{\sqrt{B}} \right). \tag{73}$$

Proof. The recursion (72) is immediate from the definitions. Using (61),

$$\mathbb{E}\big[\|E_t\|_F^2 \mid \mathcal{F}_t\big] \leqslant \mu^2 \|E_{t-1}\|_F^2 + (1-\mu)^2 \frac{\sigma^2}{B}.$$

Solving this recursion and applying Jensen gives

$$\mathbb{E}\|E_t\|_F \leqslant \mu^t \frac{\sigma}{\sqrt{B}} + \sqrt{\frac{1-\mu}{1+\mu}} \frac{\sigma}{\sqrt{B}}.$$

Since $\|A\|_* \leqslant \sqrt{r}\,\|A\|_F$, averaging over $t = 0, \ldots, T-1$ yields (73). □

Lemma C.4 (Tracking bound under spectral smoothness)**.** *Assume (62) and (64). Then*

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\|g_t - M_t\|_* \leqslant \frac{\mu L_S \eta}{1-\mu} + \frac{2\mu L_S D_S \log(eT)}{(1-\mu)T} + \sqrt{r} \left(\frac{\sigma}{(1-\mu)T\sqrt{B}} + \sqrt{\frac{1-\mu}{1+\mu}} \frac{\sigma}{\sqrt{B}} \right). \tag{74}$$

Proof. Write

$$g_t - M_t = (g_t - C_t) + (C_t - M_t), \qquad A_t := \|g_t - C_t\|_*.$$

Then

$$\begin{aligned} A_t &= \|g_t - \mu C_{t-1} - (1-\mu)g_t\|_* \\ &= \mu\|g_t - C_{t-1}\|_* \\ &\leqslant \mu\|g_t - g_{t-1}\|_* + \mu A_{t-1} \\ &\leqslant \mu L_S \|Y_t - Y_{t-1}\|_{\mathrm{op}} + \mu A_{t-1} \\ &\leqslant \mu L_S \left(\frac{2D_S}{t} + \eta\right) + \mu A_{t-1}, \end{aligned}$$

where we used (69). Iterating and averaging gives

$$\frac{1}{T}\sum_{t=0}^{T-1} A_t \leqslant \frac{\mu L_S \eta}{1-\mu} + \frac{2\mu L_S D_S \log(eT)}{(1-\mu)T}.$$

Combining this with Lemma C.3 proves (74). □

Lemma C.5 (Tracking bound under Frobenius smoothness)**.** *Assume* (63) *and* (65)*. Then*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|g_t - M_t\|_* \leqslant \frac{\mu r L_F \eta}{1-\mu} + \frac{2\mu\sqrt{r}\, L_F D_F \log(eT)}{(1-\mu)T} + \sqrt{r}\left(\frac{\sigma}{(1-\mu)T\sqrt{B}} + \sqrt{\frac{1-\mu}{1+\mu}}\frac{\sigma}{\sqrt{B}}\right). \tag{75}$$

Proof. Again write

$$g_t - M_t = (g_t - C_t) + (C_t - M_t), \qquad A_t := \|g_t - C_t\|_*.$$

Then

$$\begin{aligned} A_t &\leqslant \mu\|g_t - g_{t-1}\|_* + \mu A_{t-1} \\ &\leqslant \mu\sqrt{r}\,\|g_t - g_{t-1}\|_F + \mu A_{t-1} \\ &\leqslant \mu\sqrt{r}\, L_F \|Y_t - Y_{t-1}\|_F + \mu A_{t-1} \\ &\leqslant \mu\sqrt{r}\, L_F \left(\frac{2D_F}{t} + \eta\sqrt{r}\right) + \mu A_{t-1}, \end{aligned}$$

where we used (70). Iterating and averaging gives

$$\frac{1}{T}\sum_{t=0}^{T-1} A_t \leqslant \frac{\mu r L_F \eta}{1-\mu} + \frac{2\mu\sqrt{r}\, L_F D_F \log(eT)}{(1-\mu)T}.$$

Combining with Lemma C.3 proves (75). □

C.2 Spectral-smooth analysis

Theorem C.1 (Stationarity at the Y-iterates under spectral smoothness)**.** *Assume* (61)*,* (62)*, and* (64)*. Then for every $T \geqslant 1$,*

$$\begin{aligned} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(Y_t)\|_* \leqslant{}& \frac{\Delta + 2L_S D_S^2\left(\log(eT) + \frac{\pi^2}{6}\right)}{(1-\beta)T\eta} + \frac{L_S\eta}{2(1-\beta)} + \frac{2\mu L_S \eta}{(1-\beta)(1-\mu)} \\ &+ \frac{2L_S D_S \log(eT)}{(1-\beta)T}\left(1 + \frac{2\mu}{1-\mu}\right) + \frac{2\sqrt{r}\,\sigma}{(1-\beta)\sqrt{B}}\sqrt{\frac{1-\mu}{1+\mu}} \\ &+ \frac{2\sqrt{r}\,\sigma}{(1-\beta)(1-\mu)T\sqrt{B}}. \end{aligned} \tag{76}$$

Proof. By (62),

$$f(Y_{t+1}) \leqslant f(Y_t) + \langle g_t, Y_{t+1} - Y_t\rangle + \frac{L_S}{2}\|Y_{t+1} - Y_t\|_{\mathrm{op}}^2.$$

Using Lemma C.1,

$$Y_{t+1} - Y_t = \frac{S_t}{t+1} - \eta\left(1 - \beta + \frac{\beta}{t+1}\right) P_t,$$

hence

$$f(Y_{t+1}) \leqslant f(Y_t) + \frac{1}{t+1}\langle g_t, S_t\rangle - \eta\left(1 - \beta + \frac{\beta}{t+1}\right)\langle g_t, P_t\rangle + \frac{L_S}{2}\|Y_{t+1} - Y_t\|_{\mathrm{op}}^2.$$

Now $\|g_t\|_* \leqslant L_S D_S$, $\|S_t\|_{\mathrm{op}} \leqslant 2D_S$, $\|Y_{t+1} - Y_t\|_{\mathrm{op}} \leqslant 2D_S/(t+1) + \eta$, and Lemma C.2 gives

$$\langle g_t, P_t\rangle \geqslant \|g_t\|_* - 2\|g_t - M_t\|_*.$$

Therefore

$$f(Y_{t+1}) \leqslant f(Y_t) - \eta(1-\beta)\|g_t\|_* + 2\eta\|g_t - M_t\|_* + \frac{2L_S D_S^2}{t+1} + \frac{L_S}{2}\left(\frac{2D_S}{t+1} + \eta\right)^2. \tag{77}$$

Summing (77) from $t = 0$ to $T-1$, taking expectations, and using $f(Y_T) \geqslant f^\star$, we obtain

$$\eta(1-\beta)\sum_{t=0}^{T-1}\mathbb{E}\|g_t\|_* \leqslant \Delta + 2\eta\sum_{t=0}^{T-1}\mathbb{E}\|g_t - M_t\|_* + 2L_S D_S^2\sum_{t=0}^{T-1}\frac{1}{t+1} + \frac{L_S}{2}\sum_{t=0}^{T-1}\left(\frac{2D_S}{t+1} + \eta\right)^2.$$

Using

$$\sum_{t=0}^{T-1}\frac{1}{t+1} \leqslant \log(eT), \qquad \sum_{t=0}^{T-1}\frac{1}{(t+1)^2} \leqslant \frac{\pi^2}{6},$$

and substituting Lemma C.4 yields (76). □

C.3 Frobenius-smooth analysis

Theorem C.2 (Stationarity at the Y-iterates under Frobenius smoothness)**.** *Assume* (61)*,* (63)*, and* (65)*. Then for every $T \geqslant 1$,*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta + 2L_F D_F^2\left(\log(eT) + \frac{\pi^2}{6}\right)}{(1-\beta)T\eta} + \frac{L_F r\,\eta}{2(1-\beta)} + \frac{2\mu L_F r\,\eta}{(1-\beta)(1-\mu)} + \frac{2\sqrt{r}\,L_F D_F\log(eT)}{(1-\beta)T}\left(1 + \frac{2\mu}{1-\mu}\right) + \frac{2\sqrt{r}\,\sigma}{(1-\beta)\sqrt{B}}\sqrt{\frac{1-\mu}{1+\mu}} + \frac{2\sqrt{r}\,\sigma}{(1-\beta)(1-\mu)T\sqrt{B}}. \tag{78}$$

Proof. By (63),

$$f(Y_{t+1}) \leqslant f(Y_t) + \langle g_t, Y_{t+1} - Y_t\rangle + \frac{L_F}{2}\|Y_{t+1} - Y_t\|_F^2.$$

Using Lemma C.1,

$$Y_{t+1} - Y_t = \frac{S_t}{t+1} - \eta\left(1 - \beta + \frac{\beta}{t+1}\right) P_t.$$

Hence

$$f(Y_{t+1}) \leqslant f(Y_t) + \frac{1}{t+1}\langle g_t, S_t\rangle - \eta\left(1 - \beta + \frac{\beta}{t+1}\right)\langle g_t, P_t\rangle + \frac{L_F}{2}\|Y_{t+1} - Y_t\|_F^2.$$

Since $\|g_t\|_F \leqslant L_F D_F$, $\|S_t\|_F \leqslant 2D_F$, $\|Y_{t+1} - Y_t\|_F \leqslant 2D_F/(t+1) + \eta\sqrt{r}$, and Lemma C.2 gives

$$\langle g_t, P_t\rangle \geqslant \|g_t\|_* - 2\|g_t - M_t\|_*,$$

we obtain

$$f(Y_{t+1}) \leqslant f(Y_t) - \eta(1-\beta)\|g_t\|_* + 2\eta\|g_t - M_t\|_* + \frac{2L_F D_F^2}{t+1} + \frac{L_F}{2}\left(\frac{2D_F}{t+1} + \eta\sqrt{r}\right)^2. \quad (79)$$

Summing (79) from $t = 0$ to $T-1$, taking expectations, and using $f(Y_T) \geqslant f^\star$, we get

$$\eta(1-\beta)\sum_{t=0}^{T-1}\mathbb{E}\|g_t\|_* \leqslant \Delta + 2\eta\sum_{t=0}^{T-1}\mathbb{E}\|g_t - M_t\|_* + 2L_F D_F^2\sum_{t=0}^{T-1}\frac{1}{t+1} + \frac{L_F}{2}\sum_{t=0}^{T-1}\left(\frac{2D_F}{t+1} + \eta\sqrt{r}\right)^2.$$

Using

$$\sum_{t=0}^{T-1}\frac{1}{t+1} \leqslant \log(eT), \qquad \sum_{t=0}^{T-1}\frac{1}{(t+1)^2} \leqslant \frac{\pi^2}{6},$$

and substituting Lemma C.5 yields (78). □

C.4 Corollaries with optimized hyperparameters

Corollary C.1 (Tuned rates under spectral smoothness)**.** *Assume the hypotheses of Theorem C.1, and fix $\beta \in [0,1)$.*

(i) ***Noisy case.*** *There exist choices $\mu_T \in [0,1)$ and $\eta_T > 0$ such that*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O\left(\left(\frac{\log T}{T}\right)^{1/4}\right). \quad (80)$$

(ii) ***Noiseless case.*** *If $\sigma = 0$, then there exist choices $\mu_T \in [0,1)$ and $\eta_T > 0$ such that*

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* = O\left(\sqrt{\frac{\log T}{T}}\right). \quad (81)$$

(iii) ***Log-free specialization when*** $\beta = 0$***.*** *If $\beta = 0$, then the schedule-free drift disappears, and the rates improve to*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O\left(T^{-1/4}\right) \qquad \textit{in the noisy case,} \quad (82)$$

and

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* = O\left(T^{-1/2}\right) \qquad \textit{in the noiseless case.} \quad (83)$$

Proof. Let

$$A_{T,S} := \Delta + 2L_S D_S^2\left(\log(eT) + \frac{\pi^2}{6}\right).$$

Then $A_{T,S} = O(\log T)$.

For part (i), write

$$\alpha := 1 - \mu \in (0,1].$$

Then Theorem C.1 becomes

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{1}{1-\beta}\left[\frac{A_{T,S}}{T\eta} + \frac{L_S(4-3\alpha)}{2\alpha}\eta + \frac{2L_S D_S\log(eT)}{T}\frac{2-\alpha}{\alpha} + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha}{2-\alpha}} + \frac{2\sqrt{r}\,\sigma}{\alpha T\sqrt{B}}\right]. \quad (84)$$

For fixed α, the exact optimizer in η is

$$\eta^{\star}_{T,S}(\alpha) = \sqrt{\frac{2A_{T,S}\alpha}{L_S T(4-3\alpha)}}. \tag{85}$$

Substituting (85) into (84) gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{1}{1-\beta}\left[\sqrt{\frac{2A_{T,S}L_S}{T}\cdot\frac{4-3\alpha}{\alpha}} + \frac{2L_S D_S\log(eT)}{T}\frac{2-\alpha}{\alpha} + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha}{2-\alpha}} + \frac{2\sqrt{r}\,\sigma}{\alpha T\sqrt{B}}\right]. \tag{86}$$

Choose

$$\alpha_{T,S} := \min\left\{1,\ 2\sqrt{\frac{A_{T,S}L_S B}{r\sigma^2 T}}\right\}, \qquad \mu_{T,S} := 1-\alpha_{T,S}, \qquad \eta_{T,S} := \eta^{\star}_{T,S}(\alpha_{T,S}). \tag{87}$$

This is the leading-order optimal choice obtained by balancing the first and third terms in (86). With this choice,

$$\sqrt{\frac{2A_{T,S}L_S}{T}\cdot\frac{4-3\alpha_{T,S}}{\alpha_{T,S}}} = O\left(\left(\frac{A_{T,S}}{T}\right)^{1/4}\right),$$

and

$$\frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha_{T,S}}{2-\alpha_{T,S}}} = O\left(\left(\frac{A_{T,S}}{T}\right)^{1/4}\right).$$

The remaining terms are lower order:

$$\frac{2L_S D_S\log(eT)}{T}\frac{2-\alpha_{T,S}}{\alpha_{T,S}} = O\left(\sqrt{\frac{\log T}{T}}\right), \qquad \frac{2\sqrt{r}\,\sigma}{\alpha_{T,S}T\sqrt{B}} = O\left(\frac{1}{\sqrt{T\log T}}\right).$$

Hence

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O\left(\left(\frac{A_{T,S}}{T}\right)^{1/4}\right) = O\left(\left(\frac{\log T}{T}\right)^{1/4}\right),$$

which proves part (i).

For part (ii), when $\sigma = 0$, the stochastic terms vanish from (84). The remaining bound is increasing in μ, hence minimized by

$$\mu_{T,S} = 0 \qquad \text{equivalently} \qquad \alpha_{T,S} = 1.$$

Then the exact optimizer in η is

$$\eta_{T,S} = \sqrt{\frac{2A_{T,S}}{L_S T}}. \tag{88}$$

Substituting $\mu_{T,S} = 0$ and (88) into Theorem C.1 gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* \leqslant \frac{1}{1-\beta}\left[\sqrt{\frac{2L_S A_{T,S}}{T}} + \frac{2L_S D_S\log(eT)}{T}\right].$$

Since $A_{T,S} = O(\log T)$, this implies

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* = O\left(\sqrt{\frac{\log T}{T}}\right),$$

which proves part (ii).

For part (iii), suppose $\beta = 0$. Then $Y_t = Z_t$, hence

$$S_t = Z_t - Y_t = 0 \qquad \text{for all } t,$$

and therefore

$$Y_{t+1} - Y_t = -\eta P_t.$$

Thus the harmonic-drift term disappears from both the descent and tracking arguments. Repeating the proof of Theorem C.1 with $S_t \equiv 0$ gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta}{T\eta} + \frac{L_S\eta}{2} + \frac{2\mu L_S\eta}{1-\mu} + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{1-\mu}{1+\mu}} + \frac{2\sqrt{r}\,\sigma}{(1-\mu)T\sqrt{B}}. \tag{89}$$

Writing $\alpha := 1-\mu$, this becomes

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta}{T\eta} + \frac{L_S(4-3\alpha)}{2\alpha}\eta + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha}{2-\alpha}} + \frac{2\sqrt{r}\,\sigma}{\alpha T\sqrt{B}}.$$

For fixed α, the exact optimizer in η is

$$\eta^\star = \sqrt{\frac{2\Delta\,\alpha}{L_S T(4-3\alpha)}}.$$

Substituting this back and balancing the first two leading terms gives

$$\alpha_T \asymp \min\left\{1,\ 2\sqrt{\frac{\Delta L_S B}{r\sigma^2 T}}\right\}, \qquad \mu_T := 1-\alpha_T, \qquad \eta_T := \sqrt{\frac{2\Delta\,\alpha_T}{L_S T(4-3\alpha_T)}}.$$

With this choice,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O(T^{-1/4}),$$

which proves the noisy claim in (82).

If in addition $\sigma = 0$, then (89) reduces to

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta}{T\eta} + \frac{L_S\eta}{2}.$$

Its exact optimizer is

$$\mu_T = 0, \qquad \eta_T = \sqrt{\frac{2\Delta}{L_S T}}.$$

Substituting yields

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* \leqslant \sqrt{\frac{2L_S\Delta}{T}} = O(T^{-1/2}),$$

which proves (83). □

Corollary C.2 (Tuned rates under Frobenius smoothness)**.** *Assume the hypotheses of Theorem C.2, and fix $\beta \in [0,1)$.*

(i) ***Noisy case.*** *There exist choices $\mu_T \in [0,1)$ and $\eta_T > 0$ such that*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O\left(\left(\frac{\log T}{T}\right)^{1/4}\right). \tag{90}$$

(ii) ***Noiseless case.*** *If $\sigma = 0$, then there exist choices $\mu_T \in [0,1)$ and $\eta_T > 0$ such that*

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* = O\left(\sqrt{\frac{\log T}{T}}\right). \tag{91}$$

(iii) ***Log-free specialization when*** $\beta = 0$. *If* $\beta = 0$, *then the schedule-free drift disappears, and the rates improve to*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O\Big(T^{-1/4}\Big) \qquad \textit{in the noisy case,} \tag{92}$$

and

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* = O\Big(T^{-1/2}\Big) \qquad \textit{in the noiseless case.} \tag{93}$$

Proof. Let

$$A_{T,F} := \Delta + 2L_F D_F^2\left(\log(eT) + \frac{\pi^2}{6}\right).$$

Then $A_{T,F} = O(\log T)$.

For part (i), write

$$\alpha := 1 - \mu \in (0, 1].$$

Then Theorem C.2 becomes

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{1}{1-\beta}\Bigg[\frac{A_{T,F}}{T\eta} + \frac{L_F r(4-3\alpha)}{2\alpha}\eta + \frac{2\sqrt{r}\,L_F D_F\log(eT)}{T}\frac{2-\alpha}{\alpha} + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha}{2-\alpha}} + \frac{2\sqrt{r}\,\sigma}{\alpha T\sqrt{B}}\Bigg]. \tag{94}$$

For fixed α, the exact optimizer in η is

$$\eta^\star_{T,F}(\alpha) = \sqrt{\frac{2A_{T,F}\alpha}{L_F r\,T(4-3\alpha)}}. \tag{95}$$

Substituting (95) into (94) gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{1}{1-\beta}\Bigg[\sqrt{\frac{2A_{T,F}L_F r}{T}\cdot\frac{4-3\alpha}{\alpha}} + \frac{2\sqrt{r}\,L_F D_F\log(eT)}{T}\frac{2-\alpha}{\alpha} + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha}{2-\alpha}} + \frac{2\sqrt{r}\,\sigma}{\alpha T\sqrt{B}}\Bigg]. \tag{96}$$

Choose

$$\alpha_{T,F} := \min\left\{1,\ 2\sqrt{\frac{A_{T,F}L_F B}{\sigma^2 T}}\right\}, \qquad \mu_{T,F} := 1-\alpha_{T,F}, \qquad \eta_{T,F} := \eta^\star_{T,F}(\alpha_{T,F}). \tag{97}$$

This is the leading-order optimal choice obtained by balancing the first and third terms in (96). With this choice,

$$\sqrt{\frac{2A_{T,F}L_F r}{T}\cdot\frac{4-3\alpha_{T,F}}{\alpha_{T,F}}} = O\left(\left(\frac{A_{T,F}}{T}\right)^{1/4}\right),$$

and

$$\frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha_{T,F}}{2-\alpha_{T,F}}} = O\left(\left(\frac{A_{T,F}}{T}\right)^{1/4}\right).$$

The remaining terms are lower order:

$$\frac{2\sqrt{r}\,L_F D_F\log(eT)}{T}\frac{2-\alpha_{T,F}}{\alpha_{T,F}} = O\left(\sqrt{\frac{\log T}{T}}\right), \qquad \frac{2\sqrt{r}\,\sigma}{\alpha_{T,F}T\sqrt{B}} = O\left(\frac{1}{\sqrt{T\log T}}\right).$$

Hence

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O\left(\left(\frac{A_{T,F}}{T}\right)^{1/4}\right) = O\left(\left(\frac{\log T}{T}\right)^{1/4}\right),$$

which proves part (i).

For part (ii), when $\sigma = 0$, the stochastic terms vanish from (94). The remaining bound is increasing in μ, hence minimized by

$$\mu_{T,F} = 0 \qquad \text{equivalently} \qquad \alpha_{T,F} = 1.$$

Then the exact optimizer in η is

$$\eta_{T,F} = \sqrt{\frac{2A_{T,F}}{L_F r\, T}}. \tag{98}$$

Substituting $\mu_{T,F} = 0$ and (98) into Theorem C.2 gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* \leqslant \frac{1}{1-\beta}\left[\sqrt{\frac{2L_F r\, A_{T,F}}{T}} + \frac{2\sqrt{r}\, L_F D_F \log(eT)}{T}\right].$$

Since $A_{T,F} = O(\log T)$, this implies

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* = O\left(\sqrt{\frac{\log T}{T}}\right),$$

which proves part (ii).

For part (iii), suppose $\beta = 0$. Then $Y_t = Z_t$, so

$$S_t = 0 \qquad \text{and} \qquad Y_{t+1} - Y_t = -\eta P_t.$$

Thus the harmonic-drift term disappears from both the descent and tracking arguments. Repeating the proof of Theorem C.2 with $S_t \equiv 0$ gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta}{T\eta} + \frac{L_F r\,\eta}{2} + \frac{2\mu L_F r\,\eta}{1-\mu} + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{1-\mu}{1+\mu}} + \frac{2\sqrt{r}\,\sigma}{(1-\mu)T\sqrt{B}}. \tag{99}$$

Writing $\alpha := 1 - \mu$, this becomes

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta}{T\eta} + \frac{L_F r(4-3\alpha)}{2\alpha}\eta + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha}{2-\alpha}} + \frac{2\sqrt{r}\,\sigma}{\alpha T\sqrt{B}}.$$

For fixed α, the exact optimizer in η is

$$\eta^\star = \sqrt{\frac{2\Delta\,\alpha}{L_F r\, T(4-3\alpha)}}.$$

Substituting this back and balancing the first two leading terms gives

$$\alpha_T \asymp \min\left\{1,\ 2\sqrt{\frac{\Delta L_F B}{\sigma^2 T}}\right\}, \qquad \mu_T := 1 - \alpha_T, \qquad \eta_T := \sqrt{\frac{2\Delta\,\alpha_T}{L_F r\, T(4-3\alpha_T)}}.$$

With this choice,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O(T^{-1/4}),$$

which proves (92).

If in addition $\sigma = 0$, then (99) reduces to

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta}{T\eta} + \frac{L_F r\,\eta}{2}.$$

Its exact optimizer is

$$\mu_T = 0, \qquad \eta_T = \sqrt{\frac{2\Delta}{L_F r\, T}}.$$

Substituting yields

$$\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla f(Y_t)\|_* \leqslant \sqrt{\frac{2L_F r\,\Delta}{T}} = O(T^{-1/2}),$$

which proves (93). □

C.5 Steady-State Analysis for Weight Decay at Z

In this subsection, we prove the boundedness and steady-state characterization lemmas stated in Section 3. For convenience, we restate the lemmas here.

C.5.1 Proof of Lemma 3.1

Lemma 3.1 (Restated). *Consider the update $Z_{t+1} \leftarrow Z_t - \eta\lambda Z_t - \hat{\eta}_t \widehat{P}_t$ and assume $0 < \eta\lambda < 1$ with $\hat{\eta} = \kappa\,\eta\sqrt{mn}/\|\widehat{P}_t\|_F$ for a constant $\kappa > 0$. Then for all t, we have*

$$\|Z_t\|_F \leqslant \|Z_0\|_F(1-\eta\lambda)^t + \frac{\kappa\sqrt{mn}}{\lambda} \implies \|Z_t\|_F \leqslant \frac{\kappa\sqrt{mn}}{\lambda} \text{ as } t \to \infty.$$

Furthermore, since X_t and Y_t are convex combinations of $\{Z_s\}_{s\leqslant t}$, they satisfy the same bound.

Proof. Triangle inequality gives

$$\|Z_{t+1}\|_F \leqslant (1-\eta\lambda)\|Z_t\|_F + \hat{\eta}_t\|\widehat{P}_t\|_F = (1-\eta\lambda)\|Z_t\|_F + \kappa\,\eta\sqrt{mn}.$$

Unrolling and bounding the geometric sum:

$$\|Z_t\|_F \leqslant (1-\eta\lambda)^t\|Z_0\|_F + \kappa\,\eta\sqrt{mn}\sum_{k=0}^{t-1}(1-\eta\lambda)^k \leqslant (1-\eta\lambda)^t\|Z_0\|_F + \frac{\kappa\sqrt{mn}}{\lambda}.$$

□

C.5.2 Proof of Lemma 3.2

Lemma 3.2 (Quasi-Steady-State of Z, Restated). *Under the same conditions as Lemma 3.1, for $\eta\lambda \ll 1$ the steady-state satisfies*

$$\lambda\rho_t = \frac{\kappa}{2}\Big[-\alpha_t + \sqrt{\alpha_t^2 + 2\eta\lambda}\Big],$$

where $\rho_t := \|Z_t\|_F/\sqrt{mn}$ is the RMS norm and $\alpha_t := \langle \widehat{P}_t, Z_t\rangle/(\|\widehat{P}_t\|_F\|Z_t\|_F)$ is the alignment.

Proof. Squaring the update $Z_{t+1} = (1-\eta\lambda)Z_t - \hat{\eta}_t\widehat{P}_t$ gives

$$\|Z_{t+1}\|_F^2 = (1-\eta\lambda)^2\|Z_t\|_F^2 - 2(1-\eta\lambda)\hat{\eta}_t\langle Z_t, \widehat{P}_t\rangle + \hat{\eta}_t^2\|\widehat{P}_t\|_F^2. \tag{100}$$

The NorMuon learning rate $\hat{\eta}_t = \kappa\,\eta\sqrt{mn}/\|\widehat{P}_t\|_F$ ensures that $\hat{\eta}_t\|\widehat{P}_t\|_F = \kappa\,\eta\sqrt{mn}$ is constant. Substituting into (100):

$$\|Z_{t+1}\|_F^2 = (1-\eta\lambda)^2\|Z_t\|_F^2 - 2\kappa\,\eta(1-\eta\lambda)\alpha_t\sqrt{mn}\|Z_t\|_F + \kappa^2\eta^2 mn.$$

Dividing by mn:

$$\rho_{t+1}^2 = (1-\eta\lambda)^2\rho_t^2 - 2\kappa\,\eta(1-\eta\lambda)\alpha_t\rho_t + \kappa^2\eta^2. \tag{101}$$

In steady state, $\rho_{t+1} = \rho_t =: \rho_t$. Rearranging (101):

$$\rho_t^2\big[1-(1-\eta\lambda)^2\big] = -2\kappa\,\eta(1-\eta\lambda)\alpha_t\rho_t + \kappa^2\eta^2.$$

For $\eta\lambda \ll 1$, we have $1-(1-\eta\lambda)^2 \approx 2\eta\lambda$ and $1-\eta\lambda \approx 1$, giving the quadratic $\lambda\rho_t^2 + \kappa\,\alpha_t\rho_t - \frac{\kappa^2}{2}\eta = 0$. Taking the positive root:

$$\lambda\rho_t = \frac{\kappa}{2}\Big[-\alpha_t + \sqrt{\alpha_t^2 + 2\eta\lambda}\Big].$$

□

D ARCHITECTURE AND HYPERPARAMETER DETAILS

This appendix provides complete details of the model architectures and hyperparameter sweeps used in our experiments. All experiments were conducted on 8× NVIDIA A100-40GB GPUs [12] using PyTorch 2 [5].

Table 2: Small and large Model architecture details.

Hyperparameter	**Small Model (125M)**	**Large Model (772M)**
Vocabulary size	50,304	50,304
Number of layers	12	36
Hidden dimension (d_{model})	768	1,280
Number of attention heads	6	10
Head dimension	128	128
MLP hidden dimension	3,072	5,120
Context length	1,024	1,024
Total parameters	∼124M	∼772M
1D parameters (embedding/head)	38,633,472	64,389,120
2D parameters (attention/MLP)	84,934,656	707,788,800
Batch size (global)	512	512
Tokens per batch	524,288	524,288

Model Architecture We use a Llama-style GPT architecture with rotary positional embeddings (RoPE), RMSNorm, and QK-normalization in attention. The MLP uses a squared ReLU activation. Weight tying is applied between the token embedding and output head.

Table 3: Training budgets in steps and tokens for each model and regime.

Model		1×	2×	4×	8×
Small (125M)	Steps	5,000	10,000	20,000	40,000
	Tokens	2.6B	5.2B	10.5B	21.0B
Large (772M)	Steps	30,000	60,000	120,000	—
	Tokens	15.7B	31.5B	62.9B	—

Training Regimes We follow Chinchilla-optimal training budgets [31], scaling the number of training steps proportionally with model size. For the small model, we evaluate four training horizons denoted 1×, 2×, 4×, and 8× Chinchilla-optimal. For the large model, we train at 1×, 2×, and 4× Chinchilla-optimal. Table 3 summarizes the training budgets. All runs use a fixed global batch size of 512 sequences (∼0.5M tokens), standard for models up to 1B parameters [9], and a fixed gradient-clipping threshold of 1.0; neither was swept.

Hyperparameter Search We conduct extensive hyperparameter sweeps for each optimizer using the 125M model. In particular, for AdamW we tuned the all hyper parameters except learning rate at 2× Chinchilla but the *learning rate was tuned for each horizon*, while for SF-AdamW we tuned at 2× Chinchilla and used the same configuration for runs up to 8× Chinchilla. Using the learning rate adjustment described in Section 2, SF-NorMuon was able to use an identical learning rate and weight decay as SF-AdamW. Table 4 summarizes the search space.

Table 5 reports the best hyperparameter configuration for each optimizer on the 125M model at 8× Chinchilla-optimal training. These configurations were used for all experiments conducted for the large model.

Remarks. We make the following observations from our hyperparameter sweeps:

Table 4: Hyperparameter search space for each optimizer. Values in **bold** indicate the best configuration found. The learning rate in bold is for AdamW corresponds to the $2\times$ training duration.

Optimizer	Hyperparameter	Values Searched
AdamW	Learning rate	{0.002, 0.004, 0.006, **0.008**, 0.01}
	β_1	{**0.9**, 0.95, 0.99, 0.995}
	β_2	{0.9, **0.95**, 0.99, 0.995}
	Weight decay λ	{0, 0.05, **0.1**}
	Warmup steps	{1000, 1500, 2000, **2500**}
	Scheduler	Cosine decay to 0
SF-AdamW	Learning rate	{0.005, **0.008**, 0.01, 0.012, 0.015}
	β_1	{0.9, **0.95**, 0.99, 0.995}
	β_2	{0.9, 0.95, **0.99**, 0.995}
	Weight decay λ	{0, **0.05**, 0.1}
	Warmup steps	{1500, **2000**, 2500}
	Decay location	{$\mathbf{y}$, z}
SF-NorMuon	Learning rate	{0.005, **0.008**, 0.01, 0.012, 0.015}
	β_1	{**0.9**, 0.95, 0.99, 0.995}
	β_2	{0.9, **0.95**, 0.99, 0.995}
	Momentum μ	{**0.8**, 0.9, 0.95, 0.99}
	Weight decay λ	{0, **0.05**, 0.1}
	Warmup steps	{1500, **2000**, 2500}
	Decay location	{y, $\mathbf{z}$}

Table 5: Best hyperparameter configurations for each optimizer on the small model (learning rate corresponds to $8\times$ Chinchilla for AdamW).

Hyperparameter	AdamW	SF-AdamW	SF-NorMuon
Learning rate	0.004	0.008	0.008
(β_1, β_2) for 1D params	(0.9, 0.95)	(0.95, 0.99)	(0.95, 0.99)
(β_1, β_2) for 2D params	(0.9, 0.95)	(0.95, 0.99)	(0.9, 0.95)
Momentum μ	—	—	0.8
Weight decay λ	0.1	0.05	0.05
Warmup steps	2500	2000	2000
Decay location	—	y	z
Scheduler	Cosine	None	None

- **Momentum parameters:** For SF-AdamW, $(\beta_1, \beta_2) = (0.95, 0.99)$ consistently outperforms the SF-AdamW default of $(0.9, 0.95)$. For SF-NorMuon, we use separate beta values for 1D and 2D parameters, with the NorMuon-specific momentum $\mu = 0.8$ providing the best results.
- **Decay location:** For SF-AdamW, applying weight decay at the y iterate (gradient evaluation point) works best. For SF-NorMuon, applying decay at the z iterate (base sequence) achieves the lowest validation loss.

Learning Rate Sweep Results Figures 6–5 show the validation loss curves across different learning rates for each optimizer. For all experiments, other hyperparameters are fixed at their optimal values from Table 5.

E Ablation of SF-NorMuon and Additional Results

We conduct ablation experiments to validate the key design choices in SF-NorMuon, compare against NorMuon with a WSD schedule and with EMA weights, and report seed-to-seed variability.

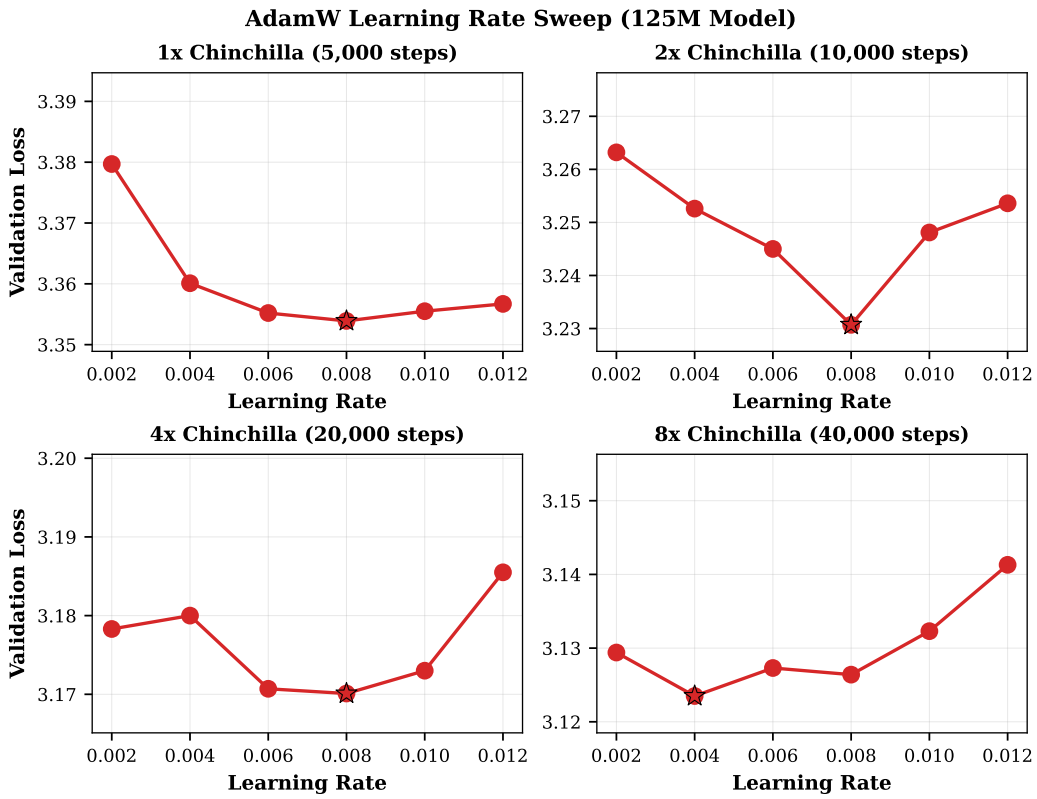

Figure 6: Learning rate sweep for AdamW with cosine scheduling on the 125M model across four training horizons: $1\times$ (5K steps), $2\times$ (10K steps), $4\times$ (20K steps), and $8\times$ (40K steps) Chinchilla-optimal. The optimal learning rate is $\eta = 0.008$ for $1\times$– $4\times$ Chinchilla, and $\eta = 0.004$ for $8\times$ Chinchilla, indicating that longer training benefits from smaller learning rates.

E.1 Ablation of SF-NorMuon Components

In Figure 3, we conduct ablation experiments to validate the two key modifications that distinguish SF-NorMuon from schedule-free spectral descent: the explicit momentum buffer and the row-wise adaptive normalization. All experiments use the 125M parameter model trained for $8\times$ Chinchilla tokens ($\approx$21B tokens) with decay at Z.

Explicit momentum. As discussed in §2, the polar update treats all singular directions uniformly, which can be more aggressive than coordinate-wise methods. We therefore introduce an explicit momentum buffer $M_t = \mu M_{t-1} + (1-\mu)G_t$ to smooth the gradient before computing its polar factor. The left panel of Figure 3 compares SF-NorMuon ($\mu = 0.8$) against the ablation with no explicit momentum ($\mu = 0$). Removing momentum increases the final validation loss from 3.14 to 3.28, a rather large degradation. This confirms that explicit momentum is essential for stable, effective training with spectral updates.

Row-wise normalization. The polar factor can produce updates with high variance across neurons, as different rows of a weight matrix may have vastly different typical magnitudes. Row-wise adaptive normalization addresses this by maintaining a running average of squared row norms and normalizing accordingly. The right panel of Figure 3 compares SF-NorMuon against the variant without row normalization on a tighter y-axis scale to reveal the small but consistent improvement. SF-NorMuon achieves a final loss of 3.136 compared to 3.142 for SF-Muon, and reaches the same loss approximately 12% faster in terms of tokens processed. This speedup is quite similar to the speedup achieved by NorMuon relative to Muon [52].

E.2 Comparison with NorMuon under a WSD Schedule

Warmup-stable-decay (WSD) schedules [32; 33] allow training to be continued from a checkpoint on the stable (constant learning rate) branch, and are therefore a natural horizon-flexible alternative to cosine decay. We train the 125M model with NorMuon under WSD, following standard configurations [22; 57] with a 300-step linear warmup and a linear cooldown over the final 20% of each target horizon. We evaluate peak learning rates $\eta = 0.008$, the best value in our cosine NorMuon sweep, and $\eta = 0.004$, motivated by prior evidence that WSD favors a peak learning rate roughly half the optimal cosine one [33]. For each learning rate, we branch from the stable checkpoints at steps 4,000 and 8,000 to obtain the $1\times$ and $2\times$ Chinchilla endpoints. Results are reported in Table 6.

Table 6: Final validation loss for the 125M model including NorMuon with a WSD schedule. WSD endpoints are obtained by cooling down from a single stable branch.

Horizon	SF-NorMuon	SF-AdamW	NorMuon (cosine)	AdamW (cosine)	NorMuon-WSD ($\eta = 0.004$)	NorMuon-WSD ($\eta = 0.008$)
$1\times$ (2.6B tokens)	3.318	3.399	**3.292**	3.354	3.310	3.317
$2\times$ (5.2B tokens)	3.229	3.282	**3.200**	3.231	3.237	3.247

At both horizons, WSD underperforms the tuned cosine schedule, consistent with the independent finding in [79] that cosine decay achieves lower final loss than WSD under separately tuned hyperparameters. At $2\times$ Chinchilla, the cooled-down WSD weights are outperformed not only by cosine NorMuon but also by SF-NorMuon. WSD is thus not a fully satisfying anytime solution: every deployable checkpoint still requires a separate cooldown branch, and, more importantly, the optimal peak learning rate itself depends on the horizon, so a value tuned for a shorter run becomes suboptimal once training is extended [57; 79]. In contrast, SF-NorMuon returns the deployable averaged iterate at every step with a single configuration.

E.3 Comparison with NorMuon and EMA Weights

A constant learning rate combined with an exponential moving average (EMA) of the weights, $\bar{W}_t = \omega \bar{W}_{t-1} + (1-\omega) W_t$, is another horizon-free alternative, and uses the same $\sim 3mn$ memory as SF-NorMuon (Remark 1). Good EMA performance requires adjusting the decay to the batch size and training duration [87], so we test the learning rates and decays recommended for our batch size in [87] on the 125M model at $1\times$ Chinchilla (Table 7).

Table 7: Final EMA validation loss for NorMuon with a constant learning rate after warmup (125M model, $1\times$ Chinchilla). For reference, cosine NorMuon reaches 3.292 and SF-NorMuon 3.318.

Learning rate	EMA decay ω	Validation loss
0.00158	0.998	3.349
0.00158	0.99	3.352
0.00316	0.998	3.346
0.00316	0.99	3.347

The best EMA configuration (3.346) is behind both cosine NorMuon and SF-NorMuon, indicating that EMA averaging alone does not remove the need for horizon-sensitive tuning in this setting.

E.4 Reproducibility Across Seeds

A single 772M run at $4\times$ Chinchilla requires 120,000 optimizer steps, making seed repetitions for every configuration prohibitively expensive. To quantify run-to-run variability, we repeat all four optimizers with two additional seeds for the 125M model at $1\times$ Chinchilla (Table 8).

The within-setting standard deviation is at most 0.006 nats, whereas the gaps between SF-NorMuon and AdamW (0.032) and between SF-NorMuon and NorMuon (0.018) are several times larger, and the ordering of the optimizers is preserved on every seed. The reported differences are therefore not artifacts of validation-loss noise.

Table 8: Validation loss for the 125M model at 1× Chinchilla across three seeds.

Optimizer	Seed 1337	Seed 1338	Seed 1339	Mean	Std.
AdamW	3.354	3.348	3.348	3.350	±0.003
NorMuon	3.292	3.305	3.304	3.300	±0.006
SF-AdamW	3.399	3.398	3.403	3.400	±0.003
SF-NorMuon	3.318	3.320	3.316	3.318	±0.002

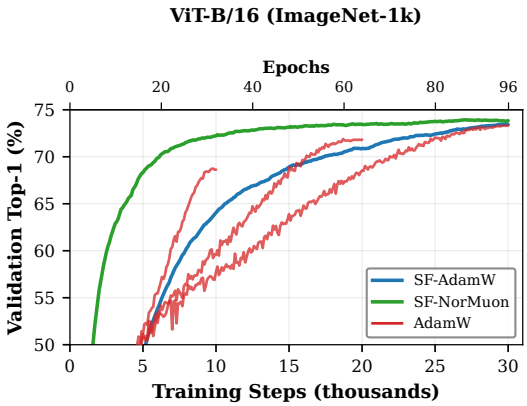

Figure 7: Best validation accuracy curve for each optimizer family on ImageNet-1K (ViT-B/16, 30K steps).

Milestone (steps)	SF-NorMuon (decay@Z)	SF-AdamW (decay@Y)	AdamW (cosine)	Speedup
10K	**72.22**	64.05	68.75	65%
20K	**73.46**	70.87	71.89	65%
30K	**73.84**	73.44	73.32	40%

Table 9: **Top-1 validation accuracy (%) at training milestones on ImageNet-1K (ViT-B/16, 30K steps).** The best accuracy at each milestone is shown in **bold**. *Speedup* is the percentage of steps saved by SF-NorMuon to reach the final accuracy of SF-AdamW at that milestone.

F Training Vision Transformers

To evaluate SF-NorMuon beyond language modeling, we train a ViT-B/16 [21] on ImageNet-1K [20] for 30K steps, comparing all three optimizer families under matched conditions. The architecture and full hyperparameter search are detailed below.

Dataset and training setup. We train on ImageNet-1K [20], which contains 1.28M training images across 1,000 classes. All experiments use a global batch size of 4096 (with gradient accumulation across 8 GPUs), image resolution 224×224, label smoothing of 0.1, and standard augmentation (random resized crop and horizontal flip). Each run trains for 30K gradient steps, and we evaluate on the full ImageNet validation set every 100 steps.

Optimizer split and hyperparameter tuning. Following the same two-optimizer paradigm as the language model experiments, *opt1* (scalars, biases, class token, position embeddings, and the classification head) always uses SF-AdamW, while *opt2* (all remaining 2D matrix parameters) uses the spectral update. We sweep learning rate, weight decay, warmup, and matrix momentum for all three families; the search space and best configurations are reported below.

Results. As shown in Figure 7 and Table 9, SF-NorMuon is the strongest optimizer, reaching **73.84%** top-1 against 73.44% for SF-AdamW and 73.32% for AdamW, and it leads at every milestone without any learning rate schedule. This mirrors the language modeling findings, with weight decay at the fast iterate Z again best for SF-NorMuon. To confirm the correctness of our implementation, we exactly replicated the original ViT-B/16 recipe [21], recovering its standard 77.9% top-1 accuracy at 90K steps (following the data processing, optimizer settings, training settings and post training higher resolution fine-tuning mentioned in this paper).

Below we provide complete details of the ViT-B/16 architecture, training setup, and hyperparameter sweeps used in the ImageNet experiments. All runs were conducted on 8× NVIDIA A100 GPUs using PyTorch 2 [5] with bf16 mixed precision and `torch.compile`.

Model Architecture

Table 10: ViT-B/16 architecture details.

Hyperparameter	ViT-B/16
Image size	224×224
Patch size	16×16
Hidden dimension (d_{model})	768
Number of layers	12
Number of attention heads	12
MLP ratio	4
MLP activation	GELU
Attention	Scaled dot-product (SDPA)
Position embedding	Learned
Classification token	`[CLS]`
Normalization	Pre-norm (LayerNorm)
Classification head init	Zero
Dropout	0.1
Total parameters	86.57M

Optimizer Split As in the language model experiments, we split the parameters into two optimizer groups. *opt1* receives all scalar parameters, biases, the class token, position embeddings, and the classification head; *opt2* receives all remaining 2D matrix parameters (patch embedding convolution, attention QKV and output projections, and MLP layers). For schedule-free runs, opt1 always uses SF-AdamW regardless of which optimizer is applied to opt2. Schedule-free optimizers are placed in train mode during parameter updates and eval mode for validation; model selection uses the validation accuracy at the x iterate.

Dataset and Data Processing We use the full ImageNet-1K dataset in torchvision `ImageFolder` layout. Training augmentation consists of `RandomResizedCrop` to 224×224 (scale $(0.05, 1.0)$, bilinear interpolation) and random horizontal flip, with symmetric normalization $(x - 0.5)/0.5$. Validation uses resize to 256, center crop to 224, and the same normalization. Color jitter is disabled for all reported runs. Mixup is available in the codebase but is also disabled for the reported runs. We use label smoothing of 0.1 throughout. The global batch size is 4096 with a per-device batch size of 128, yielding 4 gradient-accumulation microsteps on 8 GPUs. Validation is performed every 100 steps on the full validation set.

Training Duration All runs in the tuning batch train for 30K gradient steps. The AdamW baseline uses linear warmup followed by cosine decay to zero; the schedule-free methods use no explicit learning rate schedule. Warmup length is part of the sweep, ranging over $\{2\text{K}, 3\text{K}, 4\text{K}\}$ steps for all three families. Validation is performed on the full ImageNet validation set every 100 steps, and schedule-free model selection uses the validation accuracy at the x iterate.

Hyperparameter Search We tune all three optimizer families with a focused 30K-step sweep, summarized in Table 11. The AdamW baseline uses a short-warmup grid over learning rate, β_2, and warmup length. The schedule-free methods sweep learning rate, weight decay, and warmup jointly, with SF-NorMuon inheriting the learning rate, weight decay, and warmup of the selected SF-AdamW parents and additionally sweeping the matrix momentum μ.

Table 12 reports the best hyperparameter configuration for each optimizer in the 30K-step batch. Notably, SF-NorMuon uses an equal learning rate for both opt1 and opt2 groups, consistent with the rescaling approach described in Section 2.

Remarks. We highlight several observations from the 30K-step ViT tuning batch:

- **Best family:** SF-NorMuon is the strongest optimizer family, reaching 73.94% best top-1 (best over the run), ahead of SF-AdamW (73.49%) and the short-warmup AdamW grid (73.32%). Both schedule-free methods match or exceed the AdamW reference at 30K steps without a learning rate schedule.

Table 11: Hyperparameter search space for each optimizer in the 30K-step ViT-B/16 tuning batch. Values in **bold** indicate the best configuration found.

Optimizer	Hyperparameter	Values Searched
AdamW	Learning rate	{**0.003**, 0.005, 0.010}
	β_1	{**0.9**}
	β_2	{0.95, **0.99**, 0.999}
	Weight decay λ	{**0.30**}
	Warmup steps	{2,000, 3,000, **4,000**}
	Scheduler	Linear warmup then cosine decay to 0
	Dropout	**0.1**
SF-AdamW	Learning rate	{0.005, 0.008, 0.010, 0.015, **0.018**, 0.020, 0.025}
	β_1	{**0.95**}
	β_2	{**0.99**}
	Weight decay λ	{**0.15**, 0.20, 0.25, 0.30}
	Warmup steps	{**2,000**, 3,000, 4,000}
	Decay location	{$\mathbf{y}$, z}
	Dropout	**0.1**
SF-NorMuon	Learning rate	{0.008, **0.015**, 0.018, 0.025}
	β_1	{**0.95**}
	β_2	{**0.99**}
	Momentum μ	{0.8, **0.9**}
	Weight decay λ	{0.15, 0.20, **0.25**}
	Warmup steps	{2,000, **4,000**}
	Decay location	{y, $\mathbf{z}$}
	Dropout	**0.1**

Table 12: Best hyperparameter configurations for each optimizer in the 30K-step ViT-B/16 tuning batch (ImageNet-1K), with the corresponding best validation top-1.

Hyperparameter	AdamW	SF-AdamW	SF-NorMuon
Learning rate η	0.003	0.018	0.015
(β_1, β_2)	(0.9, 0.99)	(0.95, 0.99)	(0.95, 0.99)
Momentum μ (opt2)	—	—	0.9
Weight decay λ	0.3	0.15	0.25
Warmup steps	4,000	2,000	4,000
Decay location	—	y	z
Scheduler	Cosine	None	None
Dropout	0.1	0.1	0.1
Label smoothing	0.1	0.1	0.1
Best val top-1 (%)	73.32	73.49	73.94

- **Learning rate:** The AdamW baseline performs best at $\eta = 0.003$, consistent with standard ViT-B/16 training recipes. Schedule-free methods benefit from substantially higher learning rates ($\eta = 0.015$ to 0.018), which is expected given the absence of a decaying schedule.
- **Weight decay and Location:** AdamW uses $\lambda = 0.3$, consistent with the original ViT training recipe [21]. Schedule-free methods prefer moderate weight decay ($\lambda = 0.15$ to 0.25). As in the language model experiments, SF-AdamW performs best with weight decay at the y iterate, while SF-NorMuon achieves the best accuracy with decay at the z iterate. This confirms that the decay location finding generalizes across modalities.

G Reference PyTorch Implementation

Below we provide a self-contained, distributed PyTorch implementation of SF-NorMuon, closely following Algorithm 1. Our implementation is based on the code from [17; 37]. Because the Newton-Schulz orthogonalization is sharded across data-parallel ranks, the spectral update adds little cost per step. Table 13 reports the mean time per optimizer step for the 125M model; SF-NorMuon incurs only $\sim 2\%$ overhead relative to AdamW.

Table 13: Mean time per optimizer step for the 125M model on 8×A100-40GB (batch size 512, sequence length 1024, eager mode, averaged over 200 steps).

Optimizer	Time / step (ms)	Overhead vs. AdamW
AdamW	769.1	—
NorMuon	771.1	+0.26%
SF-AdamW	772.4	+0.43%
SF-NorMuon	785.5	+2.13%

Implementation notes.

- The `zeropower_via_newtonschulz5` function computes an approximation to the polar factor $P = \mathrm{polar}(M)$ using 5 iterations of the Newton-Schulz method with coefficients $(a, b, c) = (3.4445, -4.7750, 2.0315)$. The function handles both tall and wide matrices by transposing when necessary. There are other alternative methods to accomplish the polar transformation which either use randomized methods [2; 1] or differ in the choice of coefficients [3].
- In addition to the live weights `p` (the gradient evaluation sequence Y_t), the optimizer maintains three state variables per parameter: `z` (the fast iterate Z_t), `v` (the row-wise second moment estimate v_t), and `mom` (the momentum buffer M_t).
- The `eval()` method computes X_t on the fly using Z_t and Y_t. The `train()` method restores `p` to the Y_t iterate for further training.
- Weight decay is applied to the fast iterate Z_t, consistent with our analysis in Section 3, and the learning rate scaling factor `eta_scale` (default 0.2) normalizes the effective step size to be comparable to Adam's RMS-normalized updates.
- Following the NorMuon [52], the Newton-Schulz orthogonalization is sharded across the data-parallel ranks which minimizes wall-clock overhead of SF-NorMuon compared to AdamW.

```python
import math, torch
import torch.distributed as dist
@torch.compile
def zeropower_via_newtonschulz5(G, steps=5, eps=1e-7):
    assert len(G.shape) == 2
    a, b, c = (3.4445, -4.7750, 2.0315)
    X = G.bfloat16(); X /= (X.norm() + eps)
    transpose = G.size(0) > G.size(1)
    if transpose: X = X.T
    for _ in range(steps):
        A = X @ X.T; B = A @ X
        X = a * X + b * B + c * A @ B
    return X.T if transpose else X

class NorMuonScheduleFree(torch.optim.Optimizer):
    def __init__(self, params, lr=0.005, betas=(0.9, 0.95), momentum=0.8,
                 eps=1e-8, weight_decay=0.1, warmup_steps=2000, eta_scale=0.2):
        params = sorted(list(params), key=lambda x: x.numel(), reverse=True)  # align gather rounds
        defaults = dict(lr=lr, betas=betas, momentum=momentum, eps=eps,
                        weight_decay=weight_decay, warmup_steps=warmup_steps,
                        eta_scale=eta_scale, k=0, train_mode=False, weight_sum=0.0)
        super().__init__(params, defaults)
    @torch.no_grad()
    def eval(self):  # X_t = (Y_t - (1-beta) Z_t) / beta
        for group in self.param_groups:
            if group["train_mode"]:
                beta = group["betas"][0]
                for p in group["params"]:
                    if "z" in self.state[p]: p.lerp_(end=self.state[p]["z"], weight=1.0 - 1.0 / beta)
                group["train_mode"] = False
    @torch.no_grad()
    def train(self):  # restore Y_t
        for group in self.param_groups:
            if not group["train_mode"]:
                beta = group["betas"][0]
                for p in group["params"]:
                    if "z" in self.state[p]: p.lerp_(end=self.state[p]["z"], weight=1.0 - beta)
                group["train_mode"] = True
    @torch.no_grad()
    def step(self, closure=None):
        loss = closure() if closure else None
        on = dist.is_available() and dist.is_initialized() and dist.get_world_size() > 1
        world, rank = (dist.get_world_size(), dist.get_rank()) if on else (1, 0)
        for group in self.param_groups:
            beta, beta2 = group["betas"]
            mu, eps, eta_scale, decay = group["momentum"], group["eps"], group["eta_scale"], group["weight_decay"]
            k, warmup_steps = group["k"], group["warmup_steps"]
            lr = group["lr"] * ((k + 1) / warmup_steps if k < warmup_steps else 1.0)
            weight = lr * lr; group["weight_sum"] += weight; ckp1 = weight / group["weight_sum"]
            params = group["params"]
            # momentum update + lazy state init (replicated on every rank)
            for p in params:
                if p.grad is None: continue
                state = self.state[p]
                if "z" not in state:
                    state["z"], state["mom"] = p.clone(), torch.zeros_like(p)
                    state["v"] = torch.zeros(p.shape[0], device=p.device, dtype=torch.float32)
                state["mom"].mul_(mu).add_(p.grad, alpha=1.0 - mu)
            # sharded polar factor: each rank does 1/world of the matrices, then all_gather
            P = [torch.empty_like(p) for p in params]
            pad = (world - len(params) % world) % world
            pp, Pp = params + [params[-1]] * pad, P + [torch.empty_like(params[-1]) for _ in range(pad)]
            for b in range(0, len(pp), world):
                i = b + rank
                if i < len(params) and pp[i].grad is not None:
                    Pp[i].copy_(zeropower_via_newtonschulz5(self.state[pp[i]]["mom"]).to(pp[i].dtype))
                else: Pp[i].zero_()
                if on: dist.all_gather(Pp[b:b + world], Pp[i])
            # schedule-free update (replicated; uses the gathered polar factor)
            for i, p in enumerate(params):
                if p.grad is None: continue
                z, v = self.state[p]["z"], self.state[p]["v"]
                v.mul_(beta2).add_((P[i] * P[i]).mean(dim=1).float(), alpha=1.0 - beta2)
                Phat = P[i] / (v.sqrt() + eps).to(P[i].dtype).unsqueeze(1)
                m, n = p.shape
                eta_hat = eta_scale * lr * math.sqrt(m * n) / Phat.float().norm().clamp_min(1e-12)
                x_t = (p - (1.0 - beta) * z) / beta  # recover X_t
                if decay != 0.0: z.mul_(1.0 - lr * decay)
                z.sub_(eta_hat * Phat)  # z is now Z_{t+1}
                p.copy_((1.0 - beta) * z + beta * ((1.0 - ckp1) * x_t + ckp1 * z))
            group["k"] = k + 1
        return loss
```